

Reinforcement Learning

Lecture 2 : Dynamic Programming

Rémy Degenne
(remy.degenne@inria.fr)



Université
de Lille

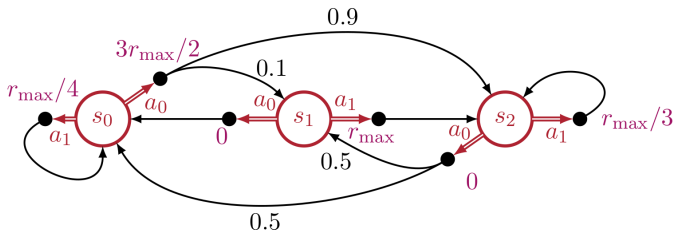


Centrale Lille, 2025/2026

Reminder : Markov Decision Process

A MDP is parameterized by a tuple $(\mathcal{S}, \mathcal{A}, R, P)$ where

- ▶ \mathcal{S} is the **state space**
- ▶ \mathcal{A} is the **action space** (or \mathcal{A}_s for each $s \in \mathcal{S}$)
- ▶ $R = (\nu_{(s,a)})_{(s,a) \in \mathcal{S} \times \mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the **reward distribution** for the state-action pair $(s, a) \rightarrow r(s, a) = \mathbb{E}_{R \sim \nu_{(s,a)}}[R]$
- ▶ $P = (p(\cdot|s, a))_{(s,a) \in \mathcal{S} \times \mathcal{A}}$ where $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the **transition kernel** associated to the state-action pair (s, a)



Reminder : Policy

A policy $\pi = (\pi_0, \pi_1, \dots)$ is a sequence of mapping $\pi_t : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ that maps a state to a distribution over actions.

Under policy π , at time t , the agent in state s_t selects

$$a_t \sim \pi_t(s_t),$$

receives the instantaneous reward

$$r_t \sim \nu_{(s_t, a_t)} \text{ such that } \mathbb{E}[r_t | s_t, a_t] = r(s_t, a_t)$$

and transits to the new state $s_{t+1} \sim p(\cdot | s_t, a_t)$.

→ a policy defines a probability model $\mathbb{P}^\pi, \mathbb{E}^\pi$ over sequences of observations :

$$s_0, a_0, r_0, s_1, a_1, r_1, \dots$$

Reminder : Value Function

Definition

The value function of a policy $\pi = (\pi_0, \pi_1, \pi_2, \dots)$ is $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$

- ① Finite-horizon criterion

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=1}^H r_t \mid s_0 = s \right]$$

- We want to compute the **optimal value** $V^*(s) = \max_{\pi} V^\pi(s)$ and an **optimal policy** π_* such that $V^* = V^{\pi_*}$.
- We will be able to do so when **\mathcal{S} and \mathcal{A} are finite**

$$S := |\mathcal{S}| < \infty \quad \text{and} \quad A := |\mathcal{A}| < \infty$$

(some optimality equation may extend to continuous state spaces)

Reminder : Value Function

Definition

The value function of a policy $\pi = (\pi_0, \pi_1, \pi_2, \dots)$ is $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$

- ① Finite-horizon criterion

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=1}^H r(s_t, a_t) \mid s_0 = s \right]$$

- We want to compute the **optimal value** $V^*(s) = \max_{\pi} V^\pi(s)$ and an **optimal policy** π_* such that $V^* = V^{\pi_*}$.
- We will be able to do so when **\mathcal{S} and \mathcal{A} are finite**

$$S := |\mathcal{S}| < \infty \quad \text{and} \quad A := |\mathcal{A}| < \infty$$

(some optimality equation may extend to continuous state spaces)

Reminder : Value Function

Definition

The value function of a policy $\pi = (\pi_0, \pi_1, \pi_2, \dots)$ is $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$

- ② Infinite horizon with a discount γ

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_0 = s \right]$$

- We want to compute the **optimal value** $V^*(s) = \max_{\pi} V^\pi(s)$ and an **optimal policy** π_* such that $V^* = V^{\pi_*}$.
- We will be able to do so when **\mathcal{S} and \mathcal{A} are finite**

$$S := |\mathcal{S}| < \infty \quad \text{and} \quad A := |\mathcal{A}| < \infty$$

(some optimality equation may extend to continuous state spaces)

Outline

1 Solving a Known MDP : Finite Horizon

2 Solving a Known MDP : the Discounted Case

- Policy Evaluation
- Computing the Optimal Policy

3 Value Iteration, Policy Iteration

Value functions

Let H be the known time horizon.

Value functions at step h

For a policy $\pi = (\pi_1, \dots, \pi_H)$,

$$V_h^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=h}^H r_t \mid s_h = s \right]$$

and

$$V_h^*(s) = \max_{\pi_h, \dots, \pi_H} \mathbb{E}^\pi \left[\sum_{t=h}^H r_t \mid s_h = s \right]$$

Goal : compute

$$V^\pi(s) = V_1^\pi(s), V^*(s) = V_1^*(s) \quad \text{and} \quad \pi^* = (\pi_1^*, \dots, \pi_H^*).$$

→ we will actually compute $V_h^\pi(s)$ and $V_h^*(s)$ for all $h \leq H$.

Value functions

Let H be the known time horizon.

Value functions at step h

For a policy $\pi = (\pi_1, \dots, \pi_H)$,

$$V_h^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=h}^H r(s_t, a_t) \mid s_h = s \right]$$

and

$$V_h^*(s) = \max_{\pi_h, \dots, \pi_H} \mathbb{E}^\pi \left[\sum_{t=h}^H r(s_t, a_t) \mid s_h = s \right]$$

Goal : compute

$$V^\pi(s) = V_1^\pi(s), V^*(s) = V_1^*(s) \quad \text{and} \quad \pi^* = (\pi_1^*, \dots, \pi_H^*).$$

→ we will actually compute $V_h^\pi(s)$ and $V_h^*(s)$ for all $h \leq H$.

Value functions

Let H be the known time horizon.

Value functions at step h

For a **deterministic** policy $\pi = (\pi_1, \dots, \pi_H)$,

$$V_h^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=h}^H r(s_t, \pi_t(s_t)) \mid s_h = s \right]$$

and

$$V_h^*(s) = \max_{\pi_h, \dots, \pi_H} \mathbb{E}^\pi \left[\sum_{t=h}^H r(s_t, \pi_t(s_t)) \mid s_h = s \right]$$

Goal : compute

$$V^\pi(s) = V_1^\pi(s), V^*(s) = V_1^*(s) \quad \text{and} \quad \pi^* = (\pi_1^*, \dots, \pi_H^*).$$

→ we will actually compute $V_h^\pi(s)$ and $V_h^*(s)$ for all $h \leq H$.

Value functions

Let H be the known time horizon.

Value functions at step h

For a policy $\pi = (\pi_1, \dots, \pi_H)$,

$$V_h^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=h}^H r_t \mid s_h = s \right]$$

and

$$V_h^*(s) = \max_{\pi_h, \dots, \pi_H} \mathbb{E}^\pi \left[\sum_{t=h}^H r_t \mid s_h = s \right]$$

How ?

- Monte-Carlo estimation ? **only approximate**
- Develop the tree of all possible realizations ? **too complex**

Bellman equations

Proposition

The value functions of a deterministic policy π satisfies the following equations : for all $h \in \{1, \dots, H\}$,

$$V_h^\pi(s) = r(s, \pi_h(s)) + \sum_{s' \in \mathcal{S}} p(s'|s, \pi_h(s)) V_{h+1}^\pi(s'),$$

with the convention that $V_{H+1}^\pi(s) = 0$ for all $s \in \mathcal{S}$.

Consequence : for a finite state space \mathcal{S} such that $|\mathcal{S}| = S$

- $V_1^\pi(s)$ can be computed using **backwards induction**
- space complexity : $S \times H$
- time complexity : $S \times (S + 1) \times H$

Bellman equations

Proposition

The value functions of a deterministic policy π satisfies the following equations : for all $h \in \{1, \dots, H\}$,

$$V_h^\pi(s) = r(s, \pi_h(s)) + \sum_{s' \in \mathcal{S}} p(s'|s, \pi_h(s)) V_{h+1}^\pi(s'),$$

with the convention that $V_{H+1}^\pi(s) = 0$ for all $s \in \mathcal{S}$.

Proof.

$$\begin{aligned} V_h^\pi(s) &= \mathbb{E}^\pi \left[r(s_h, \pi_h(s_h)) + \sum_{t=h+1}^H r(s_t, \pi_t(s_t)) \middle| s_h = s \right] \\ &= r(s, \pi_h(s)) + \mathbb{E}^\pi \left[\sum_{t=h+1}^H r(s_t, \pi_t(s_t)) \middle| s_h = s, a_h = \pi_h(s) \right] \\ &= r(s, \pi_h(s)) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s_{h+1} = s' | s_h = s, a_h = \pi_h(s)) \mathbb{E}^\pi \left[\sum_{t=h+1}^H r(s_t, \pi_t(s_t)) \middle| s_{h+1} = s' \right] \\ &= r(s, \pi_h(s)) + \sum_{s' \in \mathcal{S}} p(s'|s, \pi_h(s)) V_{h+1}^\pi(s') \end{aligned}$$

Bellman equations

Proposition

The value functions of a deterministic policy π satisfies the following equations : for all $h \in \{1, \dots, H\}$,

$$V_h^\pi(s) = r(s, \pi_h(s)) + \mathbb{E}_{s' \sim p(\cdot|s, \pi_h(s))} [V_{h+1}^\pi(s')],$$

with the convention that $V_{H+1}^\pi(s) = 0$ for all $s \in \mathcal{S}$.

These equations may be generalized :

- ▶ to a possibly infinite state space

Bellman equations

Proposition

The value functions of a policy π satisfies the following equations : for all $h \in \{1, \dots, H\}$,

$$V_h^\pi(s) = \mathbb{E}_{a \sim \pi_h(s)} \left[r(s, a) + \mathbb{E}_{s' \sim p(\cdot | s, a)} \left[V_{h+1}^\pi(s') \right] \right],$$

with the convention that $V_{H+1}^\pi(s) = 0$ for all $s \in \mathcal{S}$.

These equations may be generalized :

- ▶ to a possibly infinite state space
- ▶ to randomized policies

Bellman equations for the optimal values

Proposition

The **optimal values** V_h^* satisfy the **Bellman equations** :

$$V_h^*(s) = \max_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s'|s, a) V_{h+1}^*(s') \right] \quad \text{for all } h \leq H,$$

with the convention that $V_{H+1}^*(s) = 0$ for all $s \in \mathcal{S}$.

Moreover, an optimal policy is given by

$$\pi_h^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s'=1}^S p(s'|s, a) V_{h+1}^*(s') \right].$$

Bellman equations for the optimal values

Proposition

The **optimal values** V_h^* satisfy the **Bellman equations** :

$$V_h^*(s) = \max_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s'|s, a) V_{h+1}^*(s') \right] \quad \text{for all } h \leq H,$$

with the convention that $V_{H+1}^*(s) = 0$ for all $s \in \mathcal{S}$.

Moreover, an optimal policy is given by

$$\pi_h^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s'=1}^S p(s'|s, a) V_{h+1}^*(s') \right].$$

Consequence : for finite \mathcal{S}, \mathcal{A} such that $|\mathcal{S}| = S, |\mathcal{A}| = A$

- $\pi^* = (\pi_1^*, \dots, \pi_H^*)$ can be computed using **backwards induction**
- space complexity : $S \times H$
- time complexity : $O(S^2AH)$

Bellman equations for the optimal values

Proposition

The **optimal values** V_h^* satisfy the **Bellman equations** :

$$V_h^*(s) = \max_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s'|s, a) V_{h+1}^*(s') \right] \quad \text{for all } h \leq H,$$

with the convention that $V_{H+1}^*(s) = 0$ for all $s \in \mathcal{S}$.

Moreover, an optimal policy is given by

$$\pi_h^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s'=1}^S p(s'|s, a) V_{h+1}^*(s') \right].$$

This technique is known as **Dynamic Programming**

- ▶ term invented in the 50s by Bellman : an algorithmic principle for optimization in which solving an optimization problem of a given size reduces to solving (several) of the same optimization problem but of smaller size

Proof

$$\begin{aligned}
 V_h^*(s) &= \max_{\pi_h, \pi_{h+1}, \dots} \mathbb{E}^\pi \left[\sum_{t=h}^H r(s_t, a_t) \middle| s_h = s \right] \\
 &= \max_{\pi_h, \pi_{h+1}, \dots} \sum_{a \in \mathcal{A}} \pi_h(a_h = a | s_h = s) \mathbb{E}^\pi \left[r(s, a) + \sum_{t=h+1}^H r(s_t, a_t) \middle| s_h = s, a_h = a \right] \\
 &= \max_{\pi_h, \pi_{h+1}, \dots} \sum_{a \in \mathcal{A}} \pi_h(a_h = a | s_h = s) \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s' | s, a) \mathbb{E}^{\pi_{h+1}, \dots} \left[\sum_{t=h+1}^H r(s_t, a_t) \middle| s_{h+1} = s' \right] \right] \\
 &= \max_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s' | s, a) \max_{\pi_{h+1}, \dots} V_{h+1}^{\pi_{h+1}, \dots}(s') \right] \\
 &= \max_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s' | s, a) V_{h+1}^*(s') \right]
 \end{aligned}$$

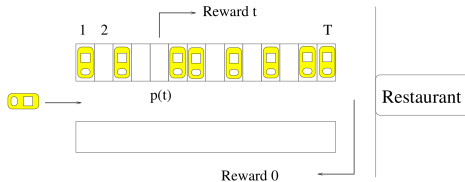
The maximizing policy is $\pi_h^*, \pi_{h+1}^*, \dots$ with

$$V_{h+1}^{\pi_{h+1}^*, \dots} = V_{h+1}^* = \operatorname{argmax}_{\pi} V_{h+1}^\pi$$

and a **deterministic** mapping $\pi_h^* : \mathcal{S} \rightarrow \mathcal{A}$ given by

$$\pi_h^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \sum_{s' \in \mathcal{S}} p(s' | s, a) V_{h+1}^*(s') \right].$$

Example : Optimal Parking



Exercise :

- model optimal parking as solving a MDP with a finite horizon
- write the optimal policy

Outline

1 Solving a Known MDP : Finite Horizon

2 Solving a Known MDP : the Discounted Case

- Policy Evaluation
- Computing the Optimal Policy

3 Value Iteration, Policy Iteration

Values functions

Let $\gamma \in (0, 1)$ be a known discount factor

Value functions

For a policy $\pi = (\pi_1, \pi_2, \dots)$,

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_1 = s \right]$$

and

$$V^*(s) = \max_{\pi} \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_1 = s \right]$$

How to compute them ?

→ We need to generalize Dynamic Programming to infinite horizon...

Values functions

Let $\gamma \in (0, 1)$ be a known discount factor

Value functions

For a **deterministic** policy $\pi = (\pi_1, \pi_2, \dots)$,

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, \pi_t(s_t)) \mid s_1 = s \right]$$

and

$$V^*(s) = \max_{\pi} \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, \pi_t(s_t)) \mid s_1 = s \right]$$

How to compute them ?

→ We need to generalize Dynamic Programming to infinite horizon...

Outline

- 1 Solving a Known MDP : Finite Horizon
- 2 Solving a Known MDP : the Discounted Case
 - Policy Evaluation
 - Computing the Optimal Policy
- 3 Value Iteration, Policy Iteration

Bellman equation for a **stationary** policy

Proposition

Any **stationary** deterministic policy π satisfies, for all $s \in \mathcal{S}$,

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^\pi(s')$$

Proof.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}^\pi \left[r(s, \pi(s)) + \sum_{t=2}^{\infty} \gamma^{t-1} r(s_t, \pi(s_t)) \middle| s_1 = s \right] \\ &= r(s, \pi(s)) + \gamma \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-2} r(s_t, \pi(s_t)) \middle| s_1 = s, a_1 = \pi(s) \right] \\ &= r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s_2 = s' | s_1 = s, a_1 = \pi(s)) \mathbb{E}^\pi \left[\sum_{t=2}^{\infty} \gamma^{t-2} r(s_t, \pi(s_t)) \middle| s_2 = s' \right] \\ &= r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) \underbrace{\mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, \pi(s_t)) \middle| s_1 = s' \right]}_{V^\pi(s')} \end{aligned}$$

Bellman equation for a stationary policy

Proposition

Any stationary deterministic policy π satisfies, for all $s \in \mathcal{S}$,

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^\pi(s')$$

More general statement :

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} [r(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s, a)} [V^\pi(s')]]$$

(also applies to infinite state space and randomized policies)

Solving the Bellman equations

Fix a stationary, deterministic policy π .

Proposition

$$\forall s \in \mathcal{S}, \quad V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^\pi(s')$$

Solving the Bellman equations

Fix a stationary, deterministic policy π .

Proposition

$$\forall s \in \mathcal{S}, \quad V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^\pi(s')$$

Introducing the vectors

$$\begin{aligned} V^\pi &= (V^\pi(s))_{s=1}^S \in \mathbb{R}^S \\ r^\pi &= (r(s, \pi(s)))_{s=1}^S \in \mathbb{R}^S \end{aligned}$$

and the matrix

$$P^\pi = \left(p(s'|s, \pi(s)) \right)_{\substack{1 \leq s \leq S \\ 1 \leq s' \leq S}} \in \mathbb{R}^{S \times S},$$

the Bellman equations rewrite

$$V^\pi = r^\pi + \gamma P^\pi V^\pi$$

Solving the Bellman equations

$$V^\pi = r^\pi + \gamma P^\pi V^\pi$$

The vector $V^\pi \in \mathbb{R}^S$ satisfies

$$\begin{aligned}(I - \gamma P^\pi) V^\pi &= r^\pi \\ V^\pi &= (I - \gamma P^\pi)^{-1} r^\pi\end{aligned}$$

provided that the matrix $I - \gamma P^\pi$ is invertible.

Proposition

The eigenvalues of the *stochastic*^a matrix P^π all belong to $[0, 1]$. As a consequence, $\gamma^{-1} \notin \text{sp}(P^\pi)$ thus $I - \gamma P^\pi$ is invertible.

a. the entries in its rows sum to 1

→ V^π can be computed by inverting a $S \times S$ matrix!

An alternative :

Exploiting the Bellman operator

Definition

The **Bellman operator** associated to a policy π is defined by

$$\begin{aligned} T^\pi : \mathbb{R}^S &\longrightarrow \mathbb{R}^S \\ V &\mapsto T^\pi(V) \end{aligned}$$

where

$$T^\pi(V)(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V(s')$$

The Bellman equation for policy π rewrites

$$V^\pi = T^\pi V^\pi$$

→ the vector V^π is a **fixed point** of the Bellman operator T^π

Intermezzo : Fixed Point Theorem

Definition

Let $(\mathcal{X}, |\cdot|)$ be a normed vector space.

A mapping $f : \mathcal{X} \rightarrow \mathcal{X}$ is called L -Lipschitz is

$$\forall (x, y) \in \mathcal{X}^2, |f(x) - f(y)| \leq L|x - y|.$$

If $L < 1$, f is called a **contraction**.

Banach's fixed point theorem

Let $(\mathcal{X}, |\cdot|)$ be a *complete* normed vector space and $f : \mathcal{X} \rightarrow \mathcal{X}$ be a contraction. Then

- ▶ there exists a **unique fixed point** x^* satisfying $f(x^*) = x^*$
- ▶ for every $x_0 \in \mathcal{X}$, the sequence defined by $x_{n+1} = f(x_n)$ for all $n \geq 0$ satisfies

$$\lim_{n \rightarrow \infty} x_n = x_*$$

Exploiting the Bellman Operator

Proposition

The operator

$$\begin{aligned} T^\pi: (\mathbb{R}^S, \|\cdot\|_\infty) &\longrightarrow (\mathbb{R}^S, \|\cdot\|_\infty) \\ V &\mapsto T^\pi(V)(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V(s') \end{aligned}$$

is a γ -contraction.

Proof.

$$\begin{aligned} \|T^\pi(V) - T^\pi(V')\|_\infty &= \sup_{s \in S} |T^\pi(V)(s) - T^\pi(V')(s)| \\ &= \sup_{s \in S} \left| \gamma \sum_{s' \in S} p(s'|s, \pi(s)) (V(s') - V'(s')) \right| \\ &\leq \gamma \sum_{s' \in S} p(s'|s, \pi(s)) \|V - V'\|_\infty \\ &= \gamma \|V - V'\|_\infty. \end{aligned}$$

Exploiting the Bellman Operator

Proposition

The operator

$$\begin{aligned} T^\pi : (\mathbb{R}^S, \|\cdot\|_\infty) &\longrightarrow (\mathbb{R}^S, \|\cdot\|_\infty) \\ V &\mapsto T^\pi(V)(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V(s') \end{aligned}$$

is a γ -contraction.

Consequence :

- ▶ V^π is the unique fixed point of T^π
- ▶ V^π can be approximated by an iterative scheme :

$$V^\pi = \lim_{k \rightarrow \infty} V_k$$

where

$$\begin{cases} V_0 & \in \mathbb{R} \\ V_{k+1} & = T^\pi(V_k) \text{ for all } k \geq 0. \end{cases}$$

Summary : Policy Evaluation

Two methods for computing $V^\pi(s)$ for all s :

- ▶ solving linear equations (**matrix inversion**)
- ▶ **iterating the Bellman operator T^π**

Other possibility : Monte-Carlo simulation

- provides only an **approximation**
- ... but **doesn't require the knowledge of $r(s, a)$ and $p(\cdot|s, a)$** ...

Back to Retail Store Management

- Constant policy : $\pi(s) = \max(M - s, c)$
- Threshold policy : $\pi(s) = \mathbb{1}_{(s \leq m_1)}(m_2 - s)$

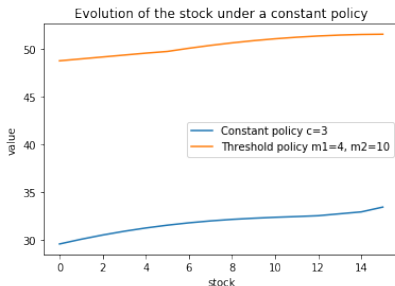


Figure – V^π for two different policies, $\gamma = 0.97$

Outline

- 1 Solving a Known MDP : Finite Horizon
- 2 Solving a Known MDP : the Discounted Case
 - Policy Evaluation
 - Computing the Optimal Policy
- 3 Value Iteration, Policy Iteration

Bellman equations for the optimal values

Proposition

$V^*(s) = \max_{\pi} V^{\pi}(s)$ satisfy the **Bellman equations** :

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

Moreover, an optimal policy is given by $\pi^* = (\pi^*, \pi^*, \dots)$ where

$$\pi^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^*(s') \right].$$

→ π^* is the **greedy policy** with respect to V^* :

Definition

The **greedy policy** with respect to a value V , $\pi = \text{greedy}(V)$ is

$$\pi(x) = \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s') \right]$$

Bellman equations for the optimal values

Proposition

$V^*(s) = \max_{\pi} V^{\pi}(s)$ satisfy the **Bellman equations** :

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

Moreover, an optimal policy is given by $\pi^* = (\pi^*, \pi^*, \dots)$ where

$$\pi^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^*(s') \right].$$

→ π^* is the **greedy policy** with respect to V^* :

Intuition : greedy(V) is the policy that “improves” a policy with value V by taking the best possible first action and then following the policy

Solving the Bellman equations

Proposition

The optimal value function V^* satisfies

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

- ▶ a system of S **non-linear** equations for computing $(V^*(s))_{s \in \mathcal{S}}$.
- no hope for a simple “matrix inversion” technique...

Bellman operator to the rescue

Optimal Bellman operator

The **optimal Bellman operator** (or dynamic programming operator) is

$$\begin{aligned} T^* : \mathbb{R}^S &\longrightarrow \mathbb{R}^S \\ V &\longmapsto T^*(V) \end{aligned}$$

where

$$T^*(V)(s) = \max_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s') \right]$$

The optimal value function satisfies

$$V^* = T^*(V^*)$$

→ V^* is a **fixed point** of the optimal Bellman operator T^* .

Optimal Bellman Operator

Properties

The optimal Bellman operator is a γ -contraction :

$$\forall V, V' \in \mathbb{R}^S, \quad \|T^*(V) - T^*(V')\|_\infty < \gamma \|V - V'\|_\infty.$$

As a consequence :

- ▶ T^* admits a unique fixed point, V^*
- ▶ for every V_0 , the sequence $V_{n+1} = T^*(V_n)$ converges to V^*

Optimal Bellman Operator

Properties

The optimal Bellman operator is a γ -contraction :

$$\forall V, V' \in \mathbb{R}^S, \quad \|T^*(V) - T^*(V')\|_\infty < \gamma \|V - V'\|_\infty.$$

As a consequence :

- ▶ T^* admits a unique fixed point, V^*
- ▶ for every V_0 , the sequence $V_{n+1} = T^*(V_n)$ converges to V^*

Proof. Uses that for all functions $| \max f - \max g | \leq \max |f - g|$.

$$\begin{aligned} \|T^*(U) - T^*(V)\|_\infty &= \max_{s \in S} \left| \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in S} p(s' | s, a) U(s') \right\} - \max_{a' \in \mathcal{A}} \left\{ r(s, a') + \gamma \sum_{s' \in S} p(s' | s, a') V(s') \right\} \right| \\ &\leq \max_{s \in S} \max_{a \in \mathcal{A}} \left| r(s, a) + \gamma \sum_{s' \in S} p(s' | s, a) U(s') - r(s, a) - \gamma \sum_{s' \in S} p(s' | s, a) V(s') \right| \\ &= \gamma \max_{s \in S} \max_{a \in \mathcal{A}} \left| \sum_{s' \in S} p(s' | s, a) (U(s') - V(s')) \right| \\ &\leq \gamma \max_{s \in S} \max_{a \in \mathcal{A}} \sum_{s' \in S} |p(s' | s, a)| \|U - V\|_\infty \leq \gamma \|U - V\|_\infty \end{aligned}$$

Optimal Bellman Operator

Properties

The optimal Bellman operator is a γ -contraction :

$$\forall V, V' \in \mathbb{R}^S, \quad \|T^*(V) - T^*(V')\|_\infty < \gamma \|V - V'\|_\infty.$$

As a consequence :

- ▶ T^* admits a unique fixed point, V^*
- ▶ for every V_0 , the sequence $V_{n+1} = T^*(V_n)$ converges to V^*

→ provides a method for approximating V^*

Outline

- 1 Solving a Known MDP : Finite Horizon
- 2 Solving a Known MDP : the Discounted Case
 - Policy Evaluation
 - Computing the Optimal Policy
- 3 Value Iteration, Policy Iteration

Value Iteration

- **Idea** : Iterate the operator T^* until V doesn't change much

Algorithm 1: Value Iteration

Input : ϵ = stopping parameter

V_0 = any function (e.g. $V_0 \leftarrow 0_S$)

```
1  $V \leftarrow V_0$ 
2 while  $\|V - T^*(V)\| \geq \epsilon$  do
3    $V \leftarrow T^*(V)$ 
4 end
```

Return: $\pi = \text{greedy}(V)$

Theorem

Value iteration converges in at most $\log \left(\frac{\|T^*(V_0) - V_0\|_\infty}{\epsilon} \right) / \log(1/\gamma)$ iterations and outputs a policy π satisfying $\|V^\pi - V^*\| \leq \frac{\gamma\epsilon}{1-\gamma}$.

Policy Iteration

- **Idea** : alternate between **policy evaluation** and **policy improvement**

Greedy policy

Recall that $\pi' = \text{greedy}(V)$ is the policy defined as

$$\pi'(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s') \right]$$

Policy improvement theorem

For any policy π , $\pi' = \text{greedy}(V^\pi)$ improves over π : $V^{\pi'} \geq V^\pi$.

Proof. uses some monotonicity property : $U \geq V \Rightarrow T^\pi(U) \geq T^\pi(V)$

- 1 by definition of the greedy policy, $T^{\pi'}(V^\pi) = T^*(V^\pi) \geq T^\pi(V^\pi) = V^\pi$
- 2 the monotonicity property yields (by induction) $(T^{\pi'})^n(V^\pi) \geq V^\pi$ for all $n \in \mathbb{N}$
- 3 using that $\lim_{n \rightarrow \infty} (T^{\pi'})^n(V^\pi) = V^{\pi'}$ concludes.

Policy Iteration

- **Idea** : alternate between **policy evaluation** and **policy improvement** .

Algorithm 2: Policy Iteration

Input : π_0 = any policy (e.g. chosen at random)

```
1  $\pi \leftarrow \pi_0$ 
2  $\pi' \leftarrow \text{NULL}$ 
3 while  $\pi \neq \pi'$  do
4    $\pi' \leftarrow \pi$ 
5   Evaluate policy  $\pi$  : compute  $V^\pi$ 
6   Improve policy  $\pi$  :  $\pi \leftarrow \text{greedy}(V^\pi)$ 
7 end
```

Return: π

Theorem

Policy iteration terminates after a **finite number of steps** and outputs the **optimal policy** π^* .

Policy Iteration

Why is that ?

Policy iteration generates a sequence of policies π_0, π_1, \dots such that

$$\pi_{k+1} = \text{greedy}(V^{\pi_k}).$$

By the policy improvement theorem,

$$V^{\pi_{k+1}} \geq V^{\pi_k}$$

and if $\pi_{k+1} \neq \pi_k$ there must exist s such that $V^{\pi_{k+1}}(s) > V^{\pi_k}(s)$
(otherwise $V^{\pi_k} = V^{\pi_{k+1}} = T^{\pi_{k+1}}(V^{\pi_{k+1}}) = T^*(V^{\pi_k})$ thus $\pi_k = \pi^*$)

→ as there is a finite number of possible values of V^π (finite number of policies), the sequence must be stationary at some point.

Implementation of VI and PI

Both algorithm can be implemented using **Q-Values** .

Definitions

The **Q-value** of a stationary policy is the expected cumulative reward when performing action a in state s and then following policy π :

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_1 = s, a_1 = a \right]$$

The **optimal Q-value** is $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) = Q^{\pi^*}(s, a)$.

Properties :

- ▶ $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$
- ▶ $V^*(s) = Q^*(s, \pi^*(s))$

Implementation of VI and PI

Both algorithm can be implemented using **Q-Values** .

Definitions

The **Q-value** of a stationary policy is the expected cumulative reward when performing action a in state s and then following policy π :

$$Q^{\pi}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi}(s')$$

The **optimal Q-value** is $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) = Q^{\pi^*}(s, a)$.

Properties :

- ▶ $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$
- ▶ $V^*(s) = Q^*(s, \pi^*(s))$

Implementation of VI and PI

Q-Values are convenient for policy improvement.

Q-value associated to a value V

To each value function V , we can associate the corresponding Q-value

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s')$$

Property : $\pi' = \text{greedy}(V)$ can be rewritten $\pi'(s) = \operatorname{argmax}_a Q(s, a)$.

Implementation of VI and PI

Q-Values are convenient for policy improvement.

Q-value associated to a value V

To each value function V , we can associate the corresponding Q-value

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s')$$

Property : $\pi' = \text{greedy}(V)$ can be rewritten $\pi'(s) = \text{argmax}_a Q(s, a)$.

Definition

The **greedy policy** with respect to a Q-value Q , $\pi = \text{greedy}(Q)$ is

$$\pi(s) = \text{argmax}_a Q(s, a)$$

Implementation of VI and PI

Q-Values are convenient for policy improvement.

Q-value associated to a value V

To each value function V , we can associate the corresponding Q-value

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s')$$

Property : $\pi' = \text{greedy}(V)$ can be rewritten $\pi' = \text{greedy}(Q)$.

Definition

The **greedy policy** with respect to a Q-value Q , $\pi = \text{greedy}(Q)$ is

$$\pi(s) = \underset{a}{\operatorname{argmax}} Q(s, a)$$

Implementation of VI and PI

Q-Values are convenient for policy improvement.

Q-value associated to a value V

To each value function V , we can associate the corresponding Q-value

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s')$$

Property : $\pi' = \text{greedy}(V)$ can be rewritten $\pi' = \text{greedy}(Q)$.

Definition

The **greedy policy** with respect to a Q-value Q , $\pi = \text{greedy}(Q)$ is

$$\pi(s) = \operatorname{argmax}_a Q(s, a)$$

Remark : $\pi^* = \text{greedy}(Q^*)$

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \text{ (apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = Q^{\pi_{k-1}}(s, a) \text{ (policy evaluation)}$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

Output : π_K

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \quad (\text{apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_{k-1}}(s')$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

Output : π_K

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \quad (\text{apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_{k-1}}(s')$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

Output : π_K

Space Complexity : $O(SA)$ in both cases

- ▶ VI : Storing Q Values + Values
- ▶ PI : Storing Q values + Policy

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \quad (\text{apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_{k-1}}(s')$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

Output : π_K

Per Iteration Time Complexity : $O(S^2A) + O(S^3)$

- ▶ VI : Compute Q values + compute S max
- ▶ PI : Compute Q values + compute S argmax + Policy Evaluation

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \quad (\text{apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_{k-1}}(s')$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

Output : π_K

Number of Iterations ? : PI often requires **few iterations**

- ▶ VI : wait for $V_{k+1} \simeq V_k$ (requires a termination criterion)
- ▶ PI : wait for $\pi_{k+1} = \pi_k$ (finite number of iterations)

Value Iteration versus Policy Iteration

In these implementations, we propose to store **Q-values**.

Value Iteration

Initialize Q_0 .

At iteration k :

$$V_k(s) = \max_a Q_{k-1}(s, a) \quad (\text{apply } T^*)$$

$$Q_k(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_k(s')$$

Output : $\pi_K(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_K(s, a)$

Policy iteration

Initialize π_0

At iteration k :

$$Q_{k-1}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_{k-1}}(s')$$

$$\pi_k(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q_{k-1}(s, a)$$

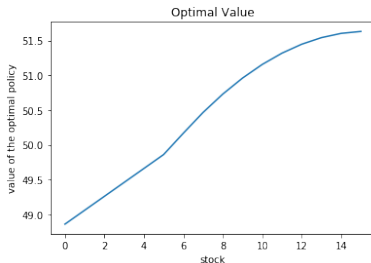
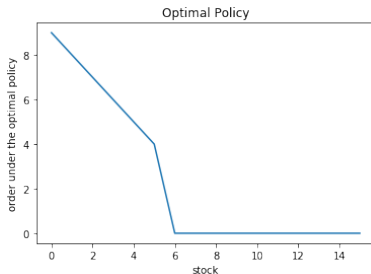
Output : π_K

Guarantees :

- ▶ VI : a policy with value very close to V^* (often π^*)
- ▶ PI : an optimal policy π^*

Back to Retail Store Management

Both VI and PI permit to find the optimal policy



π^* is a threshold policy with $m_1 = 5, m_2 = 9$
(with my choices of parameters)

Summary

We learned how to find the optimal policy in an MDP with finite state and action spaces :

- ▶ using backwards induction for a finite horizon H
- ▶ using Policy and Value iteration for an infinite horizon with a discount $\gamma \in (0, 1)$

Those two types of techniques are often indifferently referred to as

Dynamic Programming.

We are now ready to propose **reinforcement learning algorithms**, that :

- ▶ operate without the knowledge of $r(s, a)$ and $p(\cdot|s, a)$
- ▶ or in very large state spaces in which standard Dynamic Programming is intractable