Introduction to reinforcement learning

Contributors:

- Damien Ernst (dernst@uliege.be),
- Arthur Louette (arthur.uliege@uliege.be)
- Raphaël Fonteneau (Raphael.Fonteneau@uliege.be)

February 3, 2026

Outline

Reinforcement learning: problem statement and challenges

Artificial autonomous intelligent agent: formal definition

Definition (Agent)
An agent is anything that is capable of acting upon information it perceives.

Definition (Intelligent agent)
An intelligent agent is an agent capable of making decisions about how it acts based on experience, that is of learning decision from experience.

Definition (Autonomous intelligent agent)
An autonomous intelligent agent is an intelligent agent that is free to choose between different actions.

Definition (Artificial autonomous intelligent agent)

An artificial autonomous intelligent agent is anything we create that is capable of actions based on information it perceives, its own experience, and its own decisions about which actions to perform.

Since "artificial autonomous intelligent agent" is quite mouthful, we follow the convention of using "intelligent agent" or "autonomous agent" for short.

Application of intelligent agents

Intelligent agents are applied in a variety of areas: project management, electronic commerce, robotics, information retrieval, military, networking, planning and scheduling, etc.

Examples:

- A predictive maintenance agent for industrial equipment that analyzes sensor data to predict failures before they happen, scheduling maintenance only when needed and reducing downtime and costs Leroy et al. [2023].
- An autonomous delivery drone system that optimizes delivery routes and times based on traffic, weather conditions, and customer availability, learning from each delivery to improve efficiency and customer satisfaction.
- An alignment agent fine-tunes LLMs like ChatGPT, to better match user intentions. It learns from feedback to improve question interpretation and ensure accurate, relevant responses. See lecture 11 on RL and LLMs.
- A robotic harvesting assistant that navigates through orchards, using visual recognition to identify ripe fruits and vegetables. He selects products with care and precision, minimizing damage and waste. By learning from each harvest what conditions lead to the best yield and quality it helps farmers optimize picking schedules. See lecture 10 on robotic RL.

Machine learning and reinforcement learning: definitions

Definition (Machine learning)
Machine learning is a broad subfield of artificial intelligence which is concerned with the development of algorithms and techniques that allow computers to "learn".

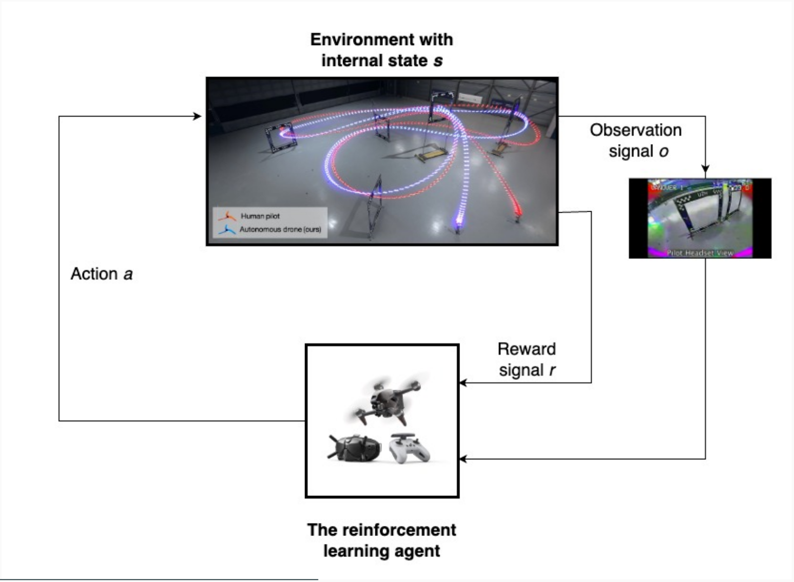
Definition (Reinforcement Learning)
Reinforcement Learning (RL in short) refers to a class of problems in machine learning which postulate an autonomous agent exploring an environment in which the agent perceives information about its current state and takes actions. The environment, in return, provides a reward signal (which can be positive or negative). The agent has as objective to maximize the (expected) cumulative reward signal over the course of the interaction.

Definition (The policy)
The policy of an agent determines the way the agent selects its action based on the information it has. A policy can be either deterministic or stochastic, either stationary or time-dependant, either Markov or history-dependant.

Research in reinforcement learning aims at designing policies which lead to large (expected) cumulative reward.

Where does the intelligence come from? The policies process in an “intelligent way” the information to select “good actions”.

An RL agent interracting with its environment



[1]

[1] Source: Kaufmann et al. [2023]

Demo

https://www.youtube.com/watch?v=yz2in2eFATE

Some generic difficulties with designing intelligent agents

- Inference problem. The environment dynamics and the mechanism behind the reward signal are (partially) unknown. The policies need to be able to infer from the information the agent has gathered from interaction with the system, “good control actions”.

- Computational complexity. The policy must be able to process the history of the observation within limited amount of computing time and memory.

- Tradeoff between exploration and exploitation.[2] To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before.

[2]May be seen as a subproblem of the general inference problem. This problem is often referred to in the “classical control theory” as the dual control problem.

The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.

- Exploring safely the environment. During an exploration phase (more generally, any phase of the agent's interaction with its environment), the agent must avoid reaching unacceptable states (e.g., states that may for example endanger its own integrity). By associating rewards of $-\infty$ to those states, exploring safely can be assimilated to a problem of exploration-exploitation.

- Adversarial environment. The environment may be adversarial. In such a context, one or several other players seek to adopt strategies that oppose the interests of the RL agent.

Problem setting and modeling assumptions

Different characterizations of RL problems

- Stochastic (e.g., $s_{t+1} = f(s_t, a_t, w_t)$ where the random disturbance w_t is drawn according to the exogenous probability distribution $P_w(\cdot)$) versus deterministic (e.g., $s_{t+1} = f(s_t, a_t)$)
- Partial observability versus full observability. The environment is said to be partially (fully) observable if the signal o_t describes partially (fully) the environment's state s_t at time t.
- Time-invariant (e.g., $s_{t+1} = f(s_t, a_t, w_t)$ with $w_t \sim P_w(\cdot)$) versus time-variant (e.g., $s_{t+1} = f(s_t, a_t, w_t, t)$) dynamics.
- Continuous (e.g., $\dot{s} = f(s, a, w)$) versus discrete dynamics (e.g., $s_{t+1} = f(s_t, a_t, w_t)$).
- Finite time versus infinite time of interaction.

- Multi-agent framework versus single-agent framework. In a multi-agent framework the environment may be itself composed of (intelligent) agents. A multi-agent framework can often be assimilated to a single-agent framework by considering that the internal states of the other agents are unobservable variables. Game theory and, more particularly, the theory of learning in games studies situations where various intelligent agents interact with each other.

- Single state versus multi-state environment. In single state environment, computation of an optimal policy for the agent is often reduced to the computation of the maximum of a stochastic function (e.g., find $a^* \in \arg\max_{a \in \mathcal{A}} \mathbb{E}_{w \sim P_w(\cdot)}[r(a, w)]$).

- Multi-objective reinforcement learning agent (reinforcement learning signal can be multi-dimensional) versus single-objective RL agent.

- Risk-adverse reinforcement learning agent. The goal of the agent is not anymore to maximize the expected cumulative reward but to maximize for example the lowest cumulative reward it could possibly obtain.

Characterization of the RL problem adopted in this class

- Dynamics of the environment: In the following, we assume that the state of the RL agent follows a discrete-time dynamics

$$s_{t+1} = f(s_t, a_t, w_t) \quad t = 0, 1, 2 \ldots$$

where $f : S \times A \times W \to S$ models the time-invariant system dynamics which outputs for any time $t \in \{0, 1, \ldots\}$, state s_t from the state space S, any action a_t from the action space A and any random disturbance w_t from the disturbance space W drawn according to a time-invariant probability distribution $w_t \sim P_w(\cdot)$, the next state $s_{t+1} \in S$.

- Reward signal:

To the transition from t to $t+1$ is associated a reward signal $\gamma^t r_t = \gamma^t r(s_t, a_t, w_t)$ where $r : S \times A \times W \to \mathbb{R}$ is a reward function supposed to be bounded by a constant $B_r > 0$ and $\gamma \in [0, 1[$ a decay (also called discount) factor.

Definition (Cumulative reward signal)

Let $h_t \in \mathcal{H}$ be the trajectory from instant time 0 to t in the combined state, action, reward spaces: $h_t = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, a_{t-1}, r_{t-1}, s_t)$. Let $\eta \in \Pi_\eta$ be a history-dependent stochastic policy and let us denote by $V^\eta(s)$ the expected return of a policy η (or expected cumulative reward signal) when the system starts from $s_0 = s$:

$$V^\eta(s) = \lim_{T \to \infty} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t \sim \eta(\cdot|h_t), w_t) | s_0 = s\right]$$

- Information available:

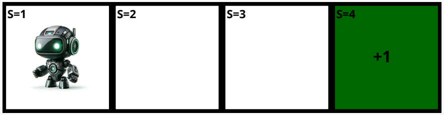
The agent does not know f, r and P_w. The only information it has on these three elements is the information contained in h_t.

For simplicity, in the following of this presentation, we assume that policies are deterministic, i.e.: $\forall \eta \in \Pi_\eta, \forall h \in \mathcal{H}, \eta(\cdot|h) = \delta(\eta(h))$ where δ denotes (here) the Dirac delta function.

Exercise 1: Computation of the cumulative reward signal

Compute the cumulative reward signal $V^{\pi}(1)$ with a constant policy $\pi(s) = 1, \quad \forall s \in \mathcal{S}$.
The state space and action space are respectively defined by:

$$\mathcal{S} = \{s \in \{1, 2, 3, 4\}\}, \quad \mathcal{A} = \{a \in \{-1, 1\}\}$$



The reward function and the dynamics are the following:

$$f(s, a) = (\min(\max(s + a, 1), 4)$$

$$r(s, a) = \begin{cases} 1 & \text{if } f(s, a) = 4 \\ 0 & \text{otherwise} \end{cases}$$

Exercise 1: Solution

Solution:

$$\begin{aligned}
V^{\pi}(1) =& \lim_{T\to\infty}[r(1,1) + \gamma r(2,1) + \gamma^2 r(3,1) + \gamma^3 r(4,1) + \ldots + \gamma^T r(4,1)] \\
=& \lim_{T\to\infty}[\gamma^2 + \ldots + \gamma^T] \\
=& \gamma^2 \lim_{T\to\infty}[1 + \ldots + \gamma^{T-2}] \\
=& \frac{\gamma^2}{1-\gamma}
\end{aligned}$$

Reminder:

$$\lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t = \frac{1}{1-\gamma}$$

Optimal control formulation of reinforcement learning

Optimal control formulation of reinforcement learning

Theorem (Optimal policy existence)

Let $\eta^* \in \Pi_\eta$ a policy such that $\forall h \in \mathcal{H}$,

$$V^{\eta^*}(s) = \max_{\eta \in \Pi_\eta} V^\eta(s) \tag{1}$$

Under some mild assumptions[3] on f, r and P_w, such a policy η^* indeed exists.

- **In reinforcement learning, we want to build policies $\hat{\eta}^*$ such that $V^{\hat{\eta}^*}$ is as close as possible (according to specific metrics) to V^{η^*}.**
- If f, r and P_w were known, we could, by putting aside the difficulty of finding in Π_η the policy η^*, design the optimal agent by solving the optimal control problem (1). However, V^η depends on f, r and P_w which are supposed to be unknown. $\Rightarrow$ How can we solve this combined inference - optimization problem?

[3] We will suppose that these mild assumptions are always satisifed afterwards.

Dynamic Programming (DP) theory reminder: optimality of stationary policies

Definition (Deterministic stationary Markov policy)

A deterministic stationary Markov control policy $\mu : \mathcal{S} \to \mathcal{A}$ selects at time t the action $a_t = \mu(s_t)$. We denote Π_μ the set of deterministic stationary Markov policies. A deterministic stationary Markov policy will output the exact same value $\mu(s_t)$ and $\mu(s_{t'})$ as soon as $s_t = s_{t'}$, whatever the history and the time.

Definition (Expected return)

The expected return of a deterministic stationary Markov policy, when the system starts from $s_0 = s$, is:

$$V^\mu(s) = \lim_{N \to \infty} \mathop{\mathbb{E}}_{w_0, w_1, \ldots, w_N} \left[\sum_{t=0}^{N} \gamma^t r(s_t, \mu(s_t), w_t) | s_0 = s \right]. \tag{2}$$

History-dependent vs stationary policies

As seen later in the course, because of the Markovian property of the state dynamics, and the fact that $\gamma < 1$, an optimal deterministic stationary Markov policy μ^* exists, satisfying:

$$V^{\mu^*}(s) = \max_{\mu \in \Pi_\mu} V^{\mu}(s), \forall s \in \mathcal{S}.$$

By denoting by η^* an optimal history-dependent policy, one can prove that

$$V^{\mu^*}(s) = V^{\eta^*}(s), \forall s \in \mathcal{S}.$$

$\Rightarrow$ **In the infinite horizon, considering only stationary Markov policies is not suboptimal.**

In the following, unless explicitly mentioned, we will consider deterministic stationary Markov policies only, and denote them using the letter μ.

Truncated return V_N^μ

Definition (Truncated return V_N^μ)

The truncated return of a stationary policy $V_N^\mu : \mathcal{S} \to \mathbb{R}$, when the system starts from $s_0 = s$, is the truncation of the N first terms of the previously defined expected return:

$$V_N^\mu(s) = \mathop{\mathbb{E}}_{w_0, w_1, \ldots, w_{N-1}} \left[\sum_{t=0}^{N-1} \gamma^t r(s_t, \mu(s_t), w_t) | s_0 = s \right], \quad \forall N \geq 1. \qquad (3)$$

with $V_0^\mu(s) \equiv 0$.

The truncated return $V_N^\mu(s)$ measures the expected sum of discounted reward over N steps under a given policy μ and starting a state s. Note that the truncated return can also be defined for a time-dependent policy π:

$$V_N^\pi(s) = \mathop{\mathbb{E}}_{w_0, w_1, \ldots, w_{N-1}} \left[\sum_{t=0}^{N-1} \gamma^t r(s_t, \pi(t, s_t), w_t) | s_0 = s \right], \quad \forall N \geq 1. \qquad (4)$$

Truncated return V_N^μ: recursive form

From equation (3), we derive the following theorem:

Theorem (Truncated return V_N^μ: recursive form)

The truncated return can be written in a recursive form:

$$V_N^\mu(s) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \left[r(s, \mu(s), w) + \gamma V_{N-1}^\mu(f(s, \mu(s), w)) \right], \quad \forall N \geq 1 \tag{5}$$

with $V_0^\mu(s) \equiv 0$.

Proof: The result is obtained by (i) isolating the first term of the sum in the definition of V_N^μ in eq. (3), (ii) putting γ outside of the sum of the remaining terms, and (iii) making a variable name change in the time index of the remaining sum to make V_{N-1}^μ appear. ■

Relation between V^μ and V_N^μ

From equations (2) and (3), we have:

$$\lim_{N\to\infty} \|V^\mu - V_N^\mu\|_\infty \to 0. \tag{6}$$

Similarly, we can also write

$$V^\mu(s) = V_N^\mu(s) + \gamma^N \mathop{\mathbb{E}}_{s_N}\left[V^\mu(s_N)\right], \quad \forall s \in \mathcal{S} \tag{7}$$

where s_N is a (random) state drawn according to the state probability distribution obtained when starting from state s and following policy μ during N steps. Assuming that the reward function is bounded by $B_r = \|r\|_\infty$ and $\gamma \in [0;1[$, we can show that there exists an upper bound on the value of $V^\mu(s)$

$$\|V^\mu\|_\infty \leq \lim_{N\to\infty} \sum_{t=0}^{N} \gamma^t B_r = \lim_{N\to\infty} \frac{1-\gamma^{N+1}}{1-\gamma} B_r = \frac{B_r}{1-\gamma} \tag{8}$$

Using (7) and (8), we can show that there exists an upper bound on $\|V^\mu - V_N^\mu\|_\infty$

$$\|V^\mu - V_N^\mu\|_\infty \leq \frac{\gamma^N}{1-\gamma} B_r. \tag{9}$$

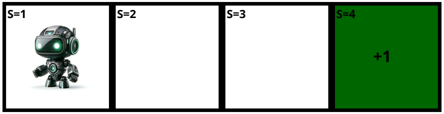
Indeed,

$$\|V^\mu - V_N^\mu\|_\infty \leq \gamma^N \|V^\mu\|_\infty = \frac{\gamma^N}{1-\gamma} B_r.$$

Exercise 2: Computation of the V_N^μ function

Compute $V_4^\mu(1)$, $V_4^\mu(2)$, $V_4^\mu(3)$ and $V_4^\mu(4)$ with policy $\pi(s) = 1, \quad \forall s \in \mathcal{S}$.
The state space and action space are respectively defined by:

$$\mathcal{S} = \{s \in \{1, 2, 3, 4\}\}, \quad \mathcal{A} = \{a \in \{-1, 1\}\}$$



The reward function and the dynamics are the following:

$$f(s, a) = (\min(\max(s + a, 1), 4)$$

$$r(s, a) = \begin{cases} 1 & \text{if } f(s, a) = 4 \\ 0 & \text{otherwise} \end{cases}$$

Exercise 2: Solution

Solution:

$$V_N^\mu(s) = r(s,\mu(s)) + \gamma V_{N-1}^\mu(f(s,\mu(s))), \quad V_0^\mu(s) \equiv 0$$

$$\begin{aligned}
&V_1^\mu(1) = 0, \quad V_1^\mu(2) = 0, \quad V_1^\mu(3) = 1, \quad V_1^\mu(4) = 1 \\
&V_2^\mu(1) = 0, \quad V_2^\mu(2) = \gamma, \quad V_2^\mu(3) = 1+\gamma, \quad V_2^\mu(4) = 1+\gamma \\
&V_3^\mu(1) = \gamma^2, \quad V_3^\mu(2) = \gamma+\gamma^2, \quad V_3^\mu(3) = V_3^\mu(4) = 1+\gamma+\gamma^2 \\
&V_4^\mu(1) = \gamma^2+\gamma^3, \quad V_4^\mu(2) = \gamma+\gamma^2+\gamma^3, \quad V_4^\mu(3) = V_4^\mu(4) = 1+\gamma+\gamma^2+\gamma^3
\end{aligned}$$

State-action value function

The V^μ function evaluates the quality of a policy μ. However, in reinforcement learning, the objective is to find policies that yield high returns. The state-action value function is a powerful tool for achieving this goal.

Definition (Q^μ-function)

Given a policy μ, we define the state-action value function $Q^\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as:

$$Q^\mu(s,a) = \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, w_t) | s_0 = s, a_0 = a, a_t = \mu(s_t) \quad \forall t \geq 1\right] \tag{10}$$

The function Q^μ evaluates the expected cumulative return of discounted reward when starting from state s, taking action a, and then following the policy μ thereafter for an infinite time horizon.

Truncated state-action value function

In the context of a finite time horizon, we define the truncated state-action value functions associated with the a policy μ:

Definition (Truncated Q_N^μ-function)

We define the functions $Q_N^\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the truncation of the N first terms of the previously defined state-action value function:

$$Q_N^\mu(s,a) = \underset{w_0,w_1,\dots,w_{N-1}}{\mathbb{E}} \left[\sum_{t=0}^{N-1} \gamma^t r(s_t,a_t,w_t) | s_0 = s, a_0 = a, a_t = \mu(s_t) \quad \forall t \geq 1 \right]. \tag{11}$$

with $Q_0^\mu(s,a) \equiv 0$.

Truncated state-action value function: recursive form

From the equation (11), we derive the following theorem:

Theorem (Truncated Q_N^μ-function: recursive form)

The Q_N^μ−functions $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ can also be written using the recurrence equation:

$$Q_N^\mu(s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}} \left[r(s,a,w) + \gamma Q_{N-1}^\mu\Big(f(s,a,w), \mu(f(s,a,w)) \Big) \right]. \tag{12}$$

with $Q_0^\mu(s,a) \equiv 0$.

Proof: The result is obtained by (i) isolating the first term of the sum in the definition of Q_N^μ in eq. (11), (ii) putting γ outside of the sum of the remaining terms, and (iii) making a variable name change in the time index of the remaining sum to make Q_{N-1}^μ appear. ■

Truncated state-action value function for time dependent policies

The truncated state-action value functions can also be defined for a time-dependent policies π:

$$Q_N^\pi(s,a) = \mathop{\mathbb{E}}_{w_0,w_1,\dots,w_{N-1}} \left[\sum_{t=0}^{N-1} \gamma^t r(s_t, a_t, w_t) | s_0 = s, a_0 = a, a_t = \pi(t, s_t) \quad \forall t \geq 1 \right]. \tag{13}$$

Truncated optimal state-action value function

Definition (Q_N-functions)

We define the functions $Q_N : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ by the recurrence equation

$$Q_N(s,a) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \left[r(s,a,w) + \gamma \max_{a' \in \mathcal{A}} Q_{N-1}(f(s,a,w),a') \right], \quad \forall N \geq 1 \tag{14}$$

with $Q_0(s,a) \equiv 0$.

Definition (N-optimal stationary policy)

A stationary policy μ_N^* is N-optimal iff it selects an optimal action when there remains exactly N steps:

$$\mu_N^*(s) \in \arg\max_{a \in \mathcal{A}} Q_N(s,a) \tag{15}$$

Finite time horizon optimal policy

Theorem (Finite time horizon optimal policy)
Let us define a time-dependent policy $\pi_N^* : \{0, \dots, N-1\} \times \mathcal{S} \to \mathcal{A}$ *as follows:*

$$\begin{aligned} \forall t \in \{0, \dots, N-1\}, \forall s \in \mathcal{S}, \pi_N^*(t, s) &= \mu_{N-t}^*(s) \\ &\in \underset{a \in \mathcal{A}}{\arg\max} Q_{N-t}(s, a) \end{aligned}$$

Choosing an action following $\pi_N^*(t, s) = \mu_{N-t}^*(s)$ *when there remain exactly* $N - t$ *steps is optimal.*

Proof: Observe that $\forall s \in \mathcal{S}, \mu_1^*(s) \in \underset{a \in \mathcal{A}}{\arg\max} \underset{w \sim P_w(\cdot)}{\mathbb{E}} [r(s, a, w)]$ and that, by induction, $\forall N \in \mathbb{N}$,

$$\forall s \in \mathcal{S}, \mu_N^*(s) \in \underset{a \in \mathcal{A}}{\arg\max} \underset{w_0, \dots, w_{N-1}}{\mathbb{E}} \left[r(s, a, w) + \sum_{t=1}^{N-1} \gamma^t r(s_t, \mu_{N-t}^*(s_t), w_t) \right] \quad (16)$$

where $s_0 = s, a_0 = a$ and $s_{t+1} = f(s_t, \mu_{N-t}^*(s_t), w_t), \forall t \in \{0, \dots, N-1\}$. ■

Finite time horizon optimal policy

Corollary

$$\forall N \in \mathbb{N}, \forall (s,a) \in \mathcal{S} \times \mathcal{A}, Q_N(s,a) = Q_N^{\pi_N^*}(s,a) \tag{17}$$

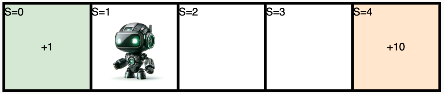
Proof: The result is obtained by replacing $\mu_{N-t}^*(\cdot)$ by $\pi_N^*(t,\cdot)$ in 16. ∎

The Q_N-functions evaluate the quality of an action a taken in state s, considering both the immediate reward and the expected discounted rewards from acting optimally thereafter, up to a finite horizon N.

A few words about finite-horizon optimality

Be careful! A N-step optimal stationary policy is not an optimal policy for a finite-time horizon. However, we can define an non-stationary optimal policy based on multiple N-step optimal stationary policies.

Example: Imagine a robot that would be able to navigate on a straight line. Starting from an initial state, the robot can either go left and get an instantaneous +1 reward, or go right 3 times in a row and get a +10 reward. As soon as the robot reaches states where it receives rewards (+1 or +10), it stays in these states and continues collecting the same reward.

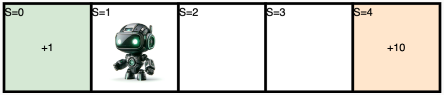


Question: depending on the value of the optimization horizon N, which direction the robot should take in order to maximize the discounted collected rewards?

Exercice 3: Optimal policy in finite time vs infinite time horizon

The state space and action space are respectively defined by:

$$\mathcal{S} = \{s \in \{0, 1, 2, 3, 4\}\}, \quad \mathcal{A} = \{a \in \{-1, 1\}\}.$$



The reward function and the dynamics are the following:

$$f(s, a) = (\min(\max(s + a, 0), 4)$$

$$r(s, a) = \begin{cases} 10 & \text{if } f(s, a) = 4 \\ 1 & \text{if } f(s, a) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Exercice 3: Solution

We notice that the optimal policy in finite time will depend on the number of remaining time steps before the time hozion N is reached:

$$\pi^*_N(t, s) = \begin{cases} 1 & \text{if } s + (N - t) \geq 4 \\ -1 & \text{otherwise.} \end{cases}$$

That means that the optimal policy is not stationary in finite time as expected.

Whereas in infinite time horizon, the optimal policy is stationary (and does does not depend on time anymore):

$$\mu^*(s) = 1 \quad \forall s \in \mathcal{S}$$

Bellman equation and convergence of Q_N

Due to their recursive definition, and following the application of the Contraction Mapping Theorem, the family of Q_N functions converges to a unique Q function. This Q function is the (unique) fixed-point of the so-called Bellman equation.

Definition (Bellman Equation and Q-function)

We define the Q-function as being the unique solution of the Bellman equation:

$$Q(s,a) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \left[r(s,a,w) + \gamma \max_{a' \in \mathcal{A}} Q(f(s,a,w),a') \right]. \tag{18}$$

Theorem (Convergence of Q_N)

The sequence of functions Q_N converges to the Q-function in the infinite norm, i.e. $\lim_{N\to\infty} \|Q_N - Q\|_\infty \to 0$.

Proof: the two above-mentioned results are direct consequence of the application of the Contraction Mapping Theorem (also referred to as the Banach fixed-point Theorem) when considering the mapping F defined as follows:

$$F(Q)(s,a) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \left[r(s,a,w) + \gamma \max_{a' \in \mathcal{A}} Q(f(s,a,w),a') \right] \tag{19}$$

and observing that $Q_{N+1}(s,a) = F(Q_N)(s,a), \forall (s,a) \in \mathcal{S} \times \mathcal{A}$. ■

The Q-function evaluates the quality of an action a in state s, considering both the immediate reward and the expected discounted rewards from acting optimally thereafter with an infinite time horizon.

Optimal stationary policy

Theorem (Optimal stationary policy)

A stationary policy μ^* defined as

$$\mu^*(s) \in \arg\max_{a \in A} Q(s,a) \tag{20}$$

is optimal in the infinite time horizon setting.

Proof: By definition, $\mu^*(s) \in \arg\max_a Q(s,a)$. Thus,

$$\begin{aligned} Q(s,a) &= \mathbb{E}_{w \sim P_w(\cdot)}\left[r(s,a,w) + \gamma Q(f(s,a,w), \mu^*(f(s,a,w))\right] && (21) \\ &= Q^{\mu^*}(s,a) && (22) \end{aligned}$$

Thus,

$$\begin{aligned} \max_a Q(s,a) &= \max_a Q^{\mu^*}(s,a) && (23) \\ &= V^{\mu^*}(s) && (24) \end{aligned}$$

Optimal stationary policy

From Equation 17 given in the previous corollary, one has:

$$\forall N \in \mathbb{N}, \forall (s,a) \in \mathcal{S} \times \mathcal{A}, Q_N(s,a) = Q_N^{\pi_N^*}(s,a) \tag{25}$$

Thus,

$$\begin{aligned} \forall N \in \mathbb{N}, \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \max_a Q_N(s,a) &= \max_a Q_N^{\pi_N^*}(s,a) \quad (26)\\ &= V_N^{\pi_N^*}(s) \quad (27) \end{aligned}$$

Taking the latter equation to the limit, since (Q_N) uniformly converge to Q one has:

$$\begin{aligned} \forall s \in \mathbf{s}, \lim_{N \to \infty} \max_a Q_N(s,a) &= \max_a Q(s,a) \quad (28)\\ &= \lim_{N \to \infty} V_N^{\pi_N^*}(s) \quad (29) \end{aligned}$$

Optimal stationary policy

From equations (24) and (29), we have that:

$$\forall s \in \mathcal{S}, V^{\mu^*}(s) = \lim_{N \to \infty} V_N^{\pi_N^*}(s) \tag{30}$$

The latter equation means that, for any state s, the value of $\mu^*(s)$ equals the asymptotical discounted return (when the horizon converges to infinity) of the optimal finite-horizon (time dependent) optimal policy.

■

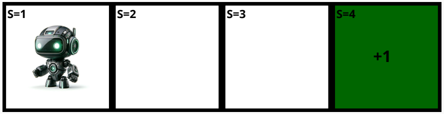
Again, be careful: necessarily, in the infinite horizon setting, the (stationary) $N-$step optimal policy μ_N^* is suboptimal with respect to the (stationary) infinite horizon policy μ^*, i.e.,

$$\forall s \in s, V^{\mu^*}(s) \geq V^{\mu_N^*}(s) \tag{31}$$

Exercise 4: Computation of the Q_N function

Compute $Q_4(s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$ with policy $\pi(s) = 1, \quad \forall s \in \mathcal{S}$.
The state space and action space are respectively defined by:

$$\mathcal{S} = \{s \in \{1,2,3,4\}\}, \quad \mathcal{A} = \{a \in \{-1,1\}\}$$



The reward function and the dynamics are the following:

$$f(s,a) = (\min(\max(s+a,1),4)$$

$$r(s,a) = \begin{cases} 1 & \text{if } f(s,a) = 4 \\ 0 & \text{otherwise} \end{cases}$$

Exercise 4: Solution

Solution:

$$Q_N(s,a) = r(s,a) + \gamma \max_{a' \in \mathcal{A}} Q_{N-1}(f(s,a),a'), \quad Q_0(s,a) \equiv 0$$

N	1	2	3	4	5
$Q_N(1,-1)$	0	0	0	γ^3	$\gamma^3+\gamma^4$
$Q_N(1,1)$	0	0	γ^2	$\gamma^2+\gamma^3$	$\gamma^2+\gamma^3+\gamma^4$
$Q_N(2,-1)$	0	0	0	γ^3	$\gamma^3+\gamma^4$
$Q_N(2,1)$	0	γ	$\gamma+\gamma^2$	$\gamma+\gamma^2+\gamma^3$	$\gamma+\gamma^2+\gamma^3+\gamma^4$
$Q_N(3,-1)$	0	0	γ^2	$\gamma^2+\gamma^3$	$\gamma^2+\gamma^3+\gamma^4$
$Q_N(3,1)$	1	$1+\gamma$	$1+\gamma+\gamma^2$	$1+\gamma+\gamma^2+\gamma^3$	$1+\gamma+\gamma^2+\gamma^3+\gamma^4$
$Q_N(4,-1)$	0	γ	$\gamma+\gamma^2$	$\gamma+\gamma^2+\gamma^3$	$\gamma+\gamma^2+\gamma^3+\gamma^4$
$Q_N(4,1)$	1	$1+\gamma$	$1+\gamma+\gamma^2$	$1+\gamma+\gamma^2+\gamma^3$	$1+\gamma+\gamma^2+\gamma^3+\gamma^4$

Bound on the suboptimality of $V^{\mu_N^*}$

Theorem (Bound on the suboptimality of $V^{\mu_N^*}$)

There exists a bound on the suboptimality of μ_N^* in comparison to μ^*, which is given by the following inequality:

$$\left\| V^{\mu^*} - V^{\mu_N^*} \right\|_\infty \leq \frac{2\gamma^N B_r}{(1-\gamma)^2} \tag{32}$$

Proof:
Let consider a stochastic environment described by the random disturbance w, a dynamics f and a reward function r:

$$\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, s_{t+1} = f(s_t, a_t, w_t) \in \mathcal{S}, \quad r_t = r(s_t, a_t, w_t) \in \mathbb{R}+, \quad w_t \sim P_w(\cdot)$$

where $\mathcal{S}$ is the state space and $\mathcal{A}$ the action space.

First, notice that the optimal stationary policy μ^* for an infinite time horizon may no longer be optimal when there remain exactly N steps.

Recall the time-dependent optimal policies $\pi^*_N : \{0, \dots, T-1\} \times \mathcal{S} \to \mathcal{A}$ defined as:

$$\begin{aligned}\forall t \in \{0, \dots, T-1\}, \forall s \in \mathcal{S}, \pi^*_N(t, s) &= \mu^*_{N-t}(s) \\ &\in \underset{a \in \mathcal{A}}{\arg\max} Q_N(s, a)\end{aligned}$$

Remember that choosing an action following μ^*_{N-t} when there remain exactly $N-t$ steps is optimal. We then define recursively

$$V_N^{\pi^*_N}(s) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \left[r(s, \mu^*_N(s), w) + \gamma V_{N-1}^{\pi^*_N}(f(s, \mu^*_N(s), w)) \right], \forall s \in \mathcal{S} \quad (33)$$

where $V_0^{\pi^*_N}(s) = 0, \forall s \in \mathcal{S}$.

Note that this policy is no longer stationary, since it depends on time. Therefore,

$$V_N^{\pi_N^*}(s) \geq V_N^{\mu^*}(s), \quad \forall s \in \mathcal{S}$$

By definition:

$$V^{\mu}(s) = \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \Big[r(s, \mu(s), w) + \gamma V^{\mu}(f(s, \mu(s), w)) \Big]$$

Since π_N^* is $N-$step optimal (cf. eq 16), we have:

$$\begin{aligned} V_N^{\pi_N^*}(s) &= \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \Big[r(s, \mu_N^*(s), w) + \gamma V_{N-1}^{\pi_N^*}(f(s, \mu_N^*(s), w)) \Big] \\ &\geq \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \Big[r(s, \mu^*(s), w) + \gamma V_{N-1}^{\pi_N^*}(f(s, \mu^*(s), w)) \Big] \end{aligned}$$

Therefore,

$$\begin{aligned} & V^{\mu^*}(s) - V^{\mu_N^*}(s) \leq \\ & V^{\mu^*}(s) - \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \Big[r(s, \mu^*(s), w) + \gamma V_{N-1}^{\pi_N^*}(f(s, \mu^*(s), w)) \Big] \\ & + \mathop{\mathbb{E}}_{w \sim P_w(\cdot)} \Big[r(s, \mu_N^*(s), w) + \gamma V_{N-1}^{\pi_N^*}(f(s, \mu_N^*(s), w)) \Big] - V^{\mu_N^*}(s) \end{aligned}$$

$$\begin{aligned} V^{\mu^*}(s) - V^{\mu^*_N}(s) \quad &\leq \quad \gamma \times \Bigg(\mathop{\mathbb{E}}_{P_w(\cdot)} \Big[V^{\mu^*}(f(s,\mu^*(s),w)) - V^{\pi^*_N}_{N-1}(f(s,\mu^*(s),w)) \Big] + \\ & \qquad \mathop{\mathbb{E}}_{P_w(\cdot)} \Big[V^{\pi^*_N}_{N-1}(f(s,\mu^*_N(s),w)) - V^{\mu^*_N}(f(s,\mu^*_N(s),w)) \Big] \Bigg) \end{aligned}$$

Taking the infinity norm, with $s'_w = f(s,\mu^*(s),w)$ and $s''_w = f(s,\mu^*_N(s),w)$,

$$\begin{aligned} \|V^{\mu^*}(s) - V^{\mu^*_N}(s)\|_\infty \leq \gamma \Bigg(& \mathbb{E}_w \left[\|V^{\mu^*}(s'_w) - V^{\pi^*_N}_{N-1}(s'_w)\|_\infty \right] \\ & + \mathbb{E}_w \left[\|V^{\pi^*_N}_{N-1}(s''_w) - V^{\mu^*_N}(s''_w)\|_\infty \right] \Bigg) \end{aligned}$$

$$\begin{aligned} \|V^{\mu^*} - V^{\mu^*_N}\|_\infty \quad &\leq \quad \gamma \|V^{\mu^*} - V^{\pi^*_N}_{N-1}\|_\infty + \gamma \|V^{\pi^*_N}_{N-1} - V^{\mu^*_N}\|_\infty \\ &\leq \quad \gamma \|V^{\mu^*} - V^{\pi^*_N}_{N-1}\|_\infty + \gamma \|V^{\pi^*_N}_{N-1} - V^{\mu^*} + V^{\mu^*} - V^{\mu^*_N}\|_\infty \\ &\leq \quad 2\gamma \|V^{\mu^*} - V^{\pi^*_N}_{N-1}\|_\infty + \gamma \|V^{\mu^*} - V^{\mu^*_N}\|_\infty \end{aligned}$$

Then,

$$\begin{aligned}(1-\gamma)\|V^{\mu^*}-V^{\mu_N^*}\|_\infty &\leq 2\gamma\|V^{\mu^*}-V_{N-1}^{\pi_N^*}\|_\infty \\ \|V^{\mu^*}-V^{\mu_N^*}\|_\infty &\leq \frac{2\gamma}{1-\gamma}\|V^{\mu^*}-V_{N-1}^{\pi_N^*}\|_\infty\end{aligned}$$

Since

$$\begin{aligned}\forall s\in\mathcal{S}, V^{\mu^*}(s)-V_{N-1}^{\pi_N^*}(s) &\leq V_{N-1}^{\mu^*}(s)+\gamma^{N-1}\lim_{K\to\infty}\sum_{k=0}^{K}\gamma^k B_r - V_{N-1}^{\pi_N^*}(s) \\ &\leq \frac{\gamma^{N-1}B_r}{1-\gamma}\end{aligned}$$

We have:

$$\begin{aligned}\|V^{\mu^*}-V^{\mu_N^*}\|_\infty &\leq \frac{2\gamma}{1-\gamma}\|V^{\mu^*}-V_{N-1}^{\pi_N^*}\|_\infty \\ &\leq \frac{2\gamma}{(1-\gamma)}\frac{\gamma^{N-1}B_r}{(1-\gamma)} \\ &\leq \frac{2\gamma^N B_r}{(1-\gamma)^2}\end{aligned}$$

■

Value Iteration

The principle of VI is to recursively apply the value update rule:

Update rule

For each state $s \in \mathcal{S}$, recursively update the value function:

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{w \sim P_W(\cdot)} \left[r(s,a,w) + \gamma V_k(f(s,a,w)) \right]$$

with $V_0(s) = 0, \forall s \in S$.

Termination: Stops when $|V_{k+1}(s) - V_k(s)| < \theta$ for all states for some predefined criterion $\theta > 0$.

Action selection

Select an action maximizing the value:

$$\mu(s) \in \arg\max_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{w \sim P_W(\cdot)} \left[r(s,a,w) + \gamma V_k(f(s,a,w)) \right]$$

Policy iteration

PI is a two-step process that alternates between evaluating a fixed policy and improving it until stability is reached.

Policy Evaluation

Compute V^μ for the current policy μ:

$$V_{k+1}^\mu(s) = \mathbb{E}_{w\sim P_W(\cdot)}[r(s,\mu(s),w) + \gamma V_k^\mu(f(s,\mu(s),w))]$$

Policy Improvement

Update the policy to be greedy w.r.t. V^μ:

$$\mu'(s) = \arg\max_{a\in\mathcal{A}} \mathbb{E}_{w\sim P_W(\cdot)}[r(s,a,w) + \gamma V^\mu(f(s,a,w))]$$

Termination: If $\mu'(s) = \mu(s)$ for all $s \in S$, the algorithm terminates.

The Value Iteration and Policy Iteration paradigms

As mentioned earlier, RL aims at designing policies which lead to large (expected) cumulative reward.

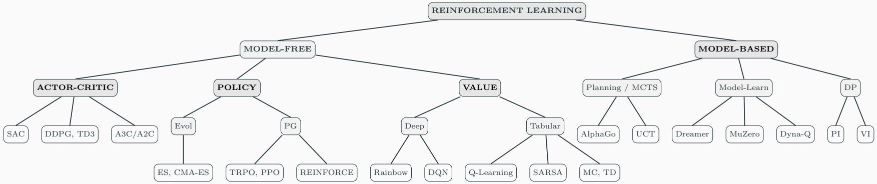
Value Iteration (VI) and Policy Iteration (PI) constitute the classical dynamic programming algorithms and form the conceptual foundation of most modern reinforcement learning methods.

While VI and PI are originally model-based approaches, they provide the mathematical blueprints for the two main branches of model-free RL as well (Value-based and Policy-based methods).

Note that model-free RL has expanded beyond just these two branches. For instances, there are also model-based RL methods that learn the model.

Actor-Critic methods blur the line between these branches, combining both value and policy components.

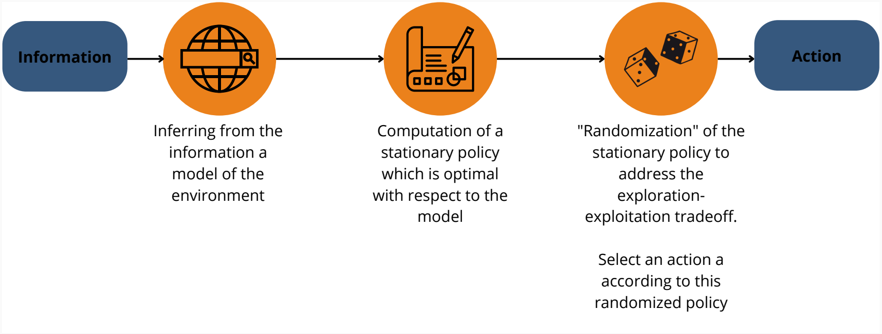
Taxonomy



Legend: DP=Dynamic Programming, VI=Value Iteration, PI=Policy Iteration, PG=Policy Gradient, (S)AC=(Soft) Actor-Critic, ES=Evolution Strategies, Evol=Evolutionary, MC=Monte Carlo, TD=Temporal Difference, SARSA=State,Action,Reward,State,Action,DQN=Deep Q Network,TRPO=Trust-Region Policy Optimization,PPO=Proximal Policy Optimization, MCTS=Monte Carlo Tree Search

Tabular reinforcement learning: planning and learning

A pragmatic model-based approach for designing good policies $\hat{\pi}^*$


Information
Inferring from the information a model of the environment
Computation of a stationary policy which is optimal with respect to the model
"Randomization" of the stationary policy to address the exploration-exploitation tradeoff.
Select an action a according to this randomized policy
Action

We focus first on to the design of functions $\hat{\pi}^*$ which realize sequentially the following three tasks:

1. "System identification" phase. Estimation from h_t of an approximate system dynamics $\hat{f}$, an approximate probability distribution $\hat{P}_w$ and an approximate reward function $\hat{r}$.

2. Resolution of the optimization problem.

Find in Π_μ a policy $\hat{\mu}^*$ such that $\forall s \in \mathcal{S}, V^{\hat{\mu}^*}(s) = \max_{\mu \in \Pi_\mu} \hat{V}^\mu(s)$
where $\hat{V}^{\hat{\mu}}$ is defined similarly as function V^μ but with $\hat{f}$, $\hat{P}_w$ and $\hat{r}$ replacing f, P_w and r, respectively.

3. Afterwards, the policy $\hat{\pi}$ selects with a probability $1 - \varepsilon(h_t)$ actions according to the policy $\hat{\mu}^*$ and with a probability $\varepsilon(h_t)$ uniformly at random among the actions. Step 3 has been introduced to address the dilemma between exploration and exploitation.[4]

[4] We will not address further the design of the 'right function' $\varepsilon : \mathcal{H} \to [0, 1]$. In many applications, it is chosen equal to a small constant (say, 0.05) everywhere.

Some algorithms for designing $\hat{\pi}^*$ when dealing with finite state-action spaces

- Until say otherwise, we consider the particular case of finite state and action spaces (i.e., $\mathcal{S} \times \mathcal{A}$ finite).

- When $\mathcal{S}$ and $\mathcal{A}$ are finite, there exists a vast panel of practical implementable RL algorithms.

- We focus first on approaches which solve separately Step 1. and Step 2. and then on approaches which solve both steps together.

- The proposed algorithms infer $\hat{\mu}^*$ from h_t. They can be adapted in a straigthforward way to episode-based reinforcement learning where a model of μ^* must be inferred from several trajectories $h_{t_1}, h_{t_2}, \ldots, h_{t_m}$ with $t_i \in \mathbb{N}_0$.

Reminder on Markov Decision Processes

Definition (Markov Decision Process)

A Markov Decision Process (MDP) is defined through the following objects: a state space $\mathcal{S}$, an action space $\mathcal{A}$, transition probabilities $p(s'|s,a)$ $\forall s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ and a reward function $r(s,a)$.

- $p(s'|s,a)$ gives the probability of reaching state s' after taking action a while being in state s.

- We consider MDPs for which we want to find decision policies that maximize the sum of reward signal $\gamma^t r(s_t, a_t)$ over an infinite time horizon.

- MDPs can be seen as a particular type of the discrete-time optimal control problem introduced earlier.

MDP Structure Definition from the System Dynamics and Reward Function

- We define[5]

$$r(s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}}[r(s,a,w)] \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{34}$$

$$p(s'|s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}}[I_{\{s'=f(s,a,w)\}}] \quad \forall s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{35}$$

- Equations (34) and (35) define the structure of an equivalent MDP in the sense that the expected return of any policy applied to the original optimal control problem is equal to its expected return in the MDP.

- The recurrence equation defining the functions Q_N can be rewritten: $Q_N(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} Q_{N-1}(s',a'), \quad \forall N \geq 1$ with $Q_0(s,a) \equiv 0$.

[5] $I_{\{logical\_expression\}} = 1$ if $logical\_expression$ is $true$ and 0 if $logical\_expression$ is $false$.

Step 1. Inferring the structure of the equivalent MDP

- Inferring a model of the environment translates into learning the parameters $p(s'|s,a)$ and $r(s,a)$ of the MDP from $h_t = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, a_{t-1}, r_{t-1}, s_t)$.
- According to equations 34 and 35, learning the dynamics and the rewards function amounts to estimate the expected value of random variables

Estimation of $r(s,a)$ and $p(s'|s,a)$:
Let $X(s,a) = \{k \in \{0,1,\dots,t-1\} | (s_k, a_k) = (s,a)\}$. Let k_1, k_2, …, $k_{\#X(s,a)}$ denote the elements of the set.[6] The values r_{k_1}, r_{k_2}, …, $r_{k_{\#X(s,a)}}$ are $\#X(s,a)$ values of the random variable $r(s,a,w)$ which are drawn independently.
Similarly, the values $I_{\{s'=s_{k_1+1}\}}$, $I_{\{s'=s_{k_2+1}\}}$, …, $I_{\{s'=s_{k_{\#X(s,a)}+1}\}}$ are $\#X(s,a)$ values of the random variable $I_{\{s'=f(s,a,w)\}}$ which are drawn independently.

It follows therefore naturally to use the sample mean as unbiased estimator of the expected values of that to estimate its mean value $r(s,a)$ and $p(s'|s,a)$:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \qquad \hat{r}(s,a) = \frac{\sum_{k \in X(s,a)} r_k}{\#X(s,a)} \tag{36}$$

$$\hat{p}(s'|s,a) = \frac{\sum_{k \in X(s,a)} I_{\{s_{k+1}=s'\}}}{\#X(s,a)} \tag{37}$$

[6] If X is a set of elements, $\#X$ denote the cardinality of X.

Reminder: statistical properties of the sample mean

- Let X_1, $X_2, \ldots$, be an infinite sequence of independent and identically distributed (i.i.d.) random variables with finite expected values $\mathbb{E}[X_i] = \mu, \forall i$.

- The weak law of large numbers states that given a collection of i.i.d. samples from a random variable with finite expected value, the sample mean converges in probability to the expected value:

$$\forall \varepsilon > 0, \lim_{n \to \infty} \Pr\left(\left| \frac{1}{n} \sum_{i=1}^{n} X_n - \mu \right| < \varepsilon \right) = 1.$$

- The strong law of large numbers states that the sample mean converges almost surely to the expected value:

$$\Pr\left(\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} X_n = \mu \right) = 1.$$

- The strong law is called "strong" because random variables which converge strongly (almost surely) are guaranteed to converge weakly (in probability). However the weak law is known to hold in certain conditions where the strong law does not hold and then the convergence is only weak (in probability).

Step 2. Computation of $\hat{\mu}^*$ identification by learning the structure of the equivalent MPD

• We compute the $\hat{Q}_N$-functions from the knowledge of $\hat{r}$ and $\hat{p}$ by exploiting the recurrence equation:

$\hat{Q}_N(s,a) = \hat{r}(s,a) + \gamma \sum_{s' \in S} \hat{p}(s'|s,a) \max_{a' \in A} Q_{N-1}(s',a'), \quad \forall N \geq 1$ with $\hat{Q}_0(s,a) \equiv 0$ and then take

$$\hat{\mu}_N^* = \arg\max_{a \in A} \hat{Q}_N(s,a) \quad \forall s \in S \tag{38}$$

as approximation of the optimal policy, with N "large enough" (e.g., right hand side of inequality (32) drops below ε).

• One can show that if the estimated MDP structure lies in an 'ε-neighborhood' of the true structure, then, $V^{\hat{\mu}^*}$ is in a '$O(\varepsilon)$-neighborhood' of V^{μ^*} where $\hat{\mu}^*(s) = \lim_{N\to\infty} \arg\max_{a \in A} \hat{Q}_N(s,a)$.

The Case of Limited Computational Resources

- Number of operations to estimate the MDP structure grows linearly with t. Memory requirements needed to store h_t also grow linearly with $t \Rightarrow$ an agent having limited computational resources will face problems after certain time of interaction.

- We describe an algorithm which requires at time t a number of operations that does not depend on t to update the MDP structure and for which the memory requirements do not grow with t:

At time 0, set $N(s,a) = 0$, $N(s,a,s') = 0$, $R(s,a) = 0$, $p(s'|s,a) = 0$, $\forall s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$.

At time $t \neq 0$, do

1. $N(s_{t-1}, a_{t-1}) \leftarrow N(s_{t-1}, a_{t-1}) + 1$
2. $N(s_{t-1}, a_{t-1}, s_t) \leftarrow N(s_{t-1}, a_{t-1}, s_t) + 1$
3. $R(s_{t-1}, a_{t-1}) \leftarrow R(s_{t-1}, a_{t-1}) + r_t$
4. $r(s_{t-1}, a_{t-1}) \leftarrow \frac{R(s_{t-1}, a_{t-1})}{N(s_{t-1}, a_{t-1})}$
5. $p(s|s_{t-1}, a_{t-1}) \leftarrow \frac{N(s_{t-1}, a_{t-1}, s)}{N(s_t, a_t)} \quad \forall s \in \mathcal{S}$

The Q-learning Algorithm

Idea: merge steps 1 and 2 to learn directly the Q-function.

The Q-learning algorithm is an algorithm that infers directly from

$$h_t = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, a_{t-1}, r_{t-1}, s_t)$$

an approximate value of the Q-function, without identifying the structure of a Markov Decision Process.

The algorithm can be described by the following steps:

1. Initialisation of $\hat{Q}_{current}(s, a)$ to 0 everywhere. Set $k = 0$.
2. $\hat{Q}_{next}(s_k, a_k) \leftarrow (1 - \alpha_k)\hat{Q}_{current}(s_k, a_k) + \alpha_k(r_k + \gamma \max_{a \in A} \hat{Q}_{current}(s_{k+1}, a))$
3. $k \leftarrow k + 1$,$\hat{Q}_{current} \leftarrow \hat{Q}_{next}$. If $k = t$, return $\hat{Q}_{current}$ and stop.

Otherwise, go back to 2.

Q-learning: some remarks

- Iteration 2. can be rewritten as $\hat{Q}_{next}(s_k, a_k) \leftarrow \hat{Q}_{current}(s_k, a_k) + \alpha_k \delta(s_k, a_k)$ where the term:

$$\delta(s_k, a_k) \quad = \quad r_k + \gamma \max_{a \in \mathcal{A}} \hat{Q}_{current}(s_{k+1}, a) - \hat{Q}_{current}(s_k, a_k), \tag{39}$$

called the *temporal difference*.

- Learning ratio α_k: The learning ratio α_k is often chosen constant with k and equal to a small value (e.g., $\alpha_k = 0.05$, $\forall k$).

- Consistency of the Q-learning algorithm: Under some particular conditions on the way α_k decreases to zero ($\lim_{t \to \infty} \sum_{k=0}^{t-1} \alpha_k \to \infty$ and $\lim_{t \to \infty} \sum_{k=0}^{t-1} \alpha_k^2 < \infty$) and the history h_t (when $t \to \infty$, every state-action pair needs to be visited an infinite number of times), $\hat{Q} \to Q$ when $t \to \infty$. (e.g. $\alpha_k = \frac{1}{k}$)

- Experience replay: At each iteration, the Q-learning algorihtm uses a sample $l_k = (s_k, a_k, r_k, s_{k+1})$ to update the function $\hat{Q}$. If rather that to use the finite sequence of sample l_0, l_2, ..., l_{t-1}, we use the infinite size sequence l_{i_1}, l_{i_2}, ... to update in a similar way $\hat{Q}$, where the i_j are i.i.d. with uniform distribution on $\{0, 2, \ldots, t-1\}$, then $\hat{Q}$ converges to the approximate Q-function computed from the estimated equivalent MDP structure.

Function approximation in reinforcement learning

Inferring $\hat{\mu}^*$ from h_t when dealing with very large or infinite state-action spaces

- Up to now, we have considered problems having discrete (and not too large) state and action spaces $\Rightarrow$ $\hat{\mu}^*$ and the $\hat{Q}_N$-functions could be represented in a tabular form.

- We consider now the case of very large or infinite state-action spaces: functions approximators need to be used to represent $\hat{\mu}^*$ and the $\hat{Q}_N$-functions.

- These function approximators need to be used in a way that there are able to 'well generalize' over the whole state-action space the information contained in h_t.

- There is a vast literature on function approximators in reinforcement learning. We focus first on one algorithm named 'fitted Q iteration' which computes the functions $\hat{Q}_N$ from h_t by solving a sequence of batch mode supervised learning problems.

Reminder: Batch mode supervised learning

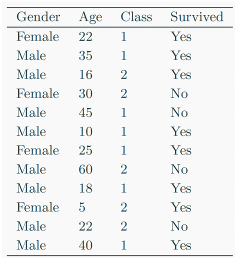
- A batch mode Supervised Learning (SL) algorithm infers from a set of input-output (input = information state); (output = class label, real number, graph, etc) a model which explains "at best" these input-output pairs.

- A loose formalisation of the SL problem: Let I be the input space, O the output space, Ξ the disturbance space. Let $g : I \times \Xi \rightarrow O$. Let $P_\xi(\cdot|i)$ a conditional probability distribution over the disturbance space.

We assume that we have a training set $\mathcal{T}S = \{(i^l, o^l)\}_{l=1}^{\#\mathcal{T}S}$ such that o^l has been generated from i^l by the following mechanism: draw $\xi \in \Xi$ according to $P_\xi(\cdot|i^l)$ and then set $o^l = g(i^l, \xi)$.

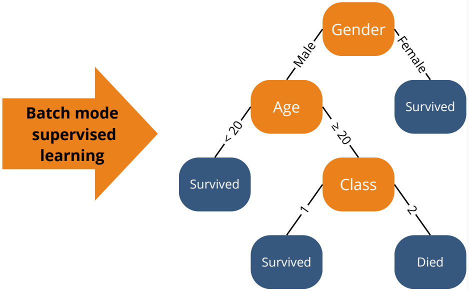
From the sole knowledge of $\mathcal{T}S$, supervised learning aims at finding a function $\hat{\bar{g}} : I \rightarrow O$ which is a 'good approximation' of the function $\bar{g}(i) = \underset{\xi \sim P_\xi(\cdot)}{\mathbb{E}}[g(i, \xi)]$

• Typical supervised learning methods are: kernel-based methods, (deep) neural networks, tree-based methods.



Gender	Age	Class	Survived
Female	22	1	Yes
Male	35	1	Yes
Male	16	2	Yes
Female	30	2	No
Male	45	1	No
Male	10	1	Yes
Female	25	1	Yes
Male	60	2	No
Male	18	1	Yes
Female	5	2	Yes
Male	22	2	No
Male	40	1	Yes

Table 1: Titanic Survival Dataset



• Supervised learning highly successful: state-of-the art SL algorithms have been successfully applied to problems where the input state was composed thousands of components.

The fitted Q iteration algorithm

- Fitted Q iteration computes from h_t the functions $\hat{Q}_1$, $\hat{Q}_2$, $\ldots$, $\hat{Q}_N$, approximations of Q_1, Q_2, $\ldots$, Q_N. At step $N > 1$, the algorithm uses the function $\hat{Q}_{N-1}$ together with h_t to compute a new training set from which a SL algorithm outputs $\hat{Q}_N$. More precisely, this iterative algorithm works as follows:

First iteration: the algorithm determines a model $\hat{Q}_1$ of $Q_1(s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}}[r(s,a,w)]$ by running a SL algorithms on the training set:

$$\mathcal{T}S = \{((s_k, a_k), r_k)\}_{k=0}^{t-1} \tag{40}$$

Motivation: One can assimilate $S \times A$ to I, $\mathbb{R}$ to O, W to Ξ, $P_w(\cdot)$ to $P_\xi(\cdot)$, $r(s,a,w)$ to $g(i,\xi)$ and $Q_1(s,a)$ to $\overline{g}$. From there, we can observe that a SL algorithm applied to the training set described by equation (40) will produce a model of Q_1.

Iteration $N > 1$: the algorithm outputs a model $\hat{Q}_N$ of $Q_N(s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}}[r(s,a,w) + \gamma \max_{a' \in \mathcal{A}} Q_{N-1}(f(s,a,w),a')]$ by running a SL algorithms on the training set:

$$\mathcal{T}S = \{((s_k,a_k), r_k + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_{N-1}(s_{k+1},a')\}_{k=0}^{t-1}$$

Motivation: One can reasonably suppose that $\hat{Q}_{N-1}$ is a a sufficiently good approximation of Q_{N-1} to be considered equal to this latter function. Assimilate $\mathcal{S} \times \mathcal{A}$ to I, $\mathbb{R}$ to O, W to Ξ, $P_w(\cdot)$ to $P_\xi(\cdot)$, $r(s,a,w)$ to $g(i,\xi)$ and $Q_N(s,a)$ to $\overline{g}$. From there, we observe that a SL algorithm applied to the training set described by equation (41) will produce a model of Q_N.

- The algorithm stops when N is 'large enough' and $\hat{\mu}^*_N(s) \in \arg\max_{a \in \mathcal{A}} \hat{Q}_N(s,a)$ is taken as approximation of $\mu^*(s)$.

The fitted Q iteration algorithm: some remarks

- Performances of the algorithm depends on the supervised learning (SL) method chosen.

- Excellent and stable performances have been observed when combined with supervised learning methods based on ensemble of regression trees and of course, with deep neural nets, especially when images are used as input.

- Fitted Q iteration algorithm can be used with any set of one-step system transitions (s_t, a_t, r_t, s_{t+1}) where each one-step system transition gives information about: a state, the action taken while being in this state, the reward signal observed and the next state reached.

- Consistency, that is convergence towards an optimal solution when the number of one-step system transitions tends to infinity, can be ensured under appropriate assumptions on the SL method, the sampling process, the system dynamics and the reward function.

Computation of $\hat{\mu}^*$: from an inference problem to a problem of computational complexity

- When having at one's disposal only a few one-step system transitions, the main problem is a problem of inference.

- Computational complexity of the fitted Q iteration algorithm grows with the number M of one-step system transitions (s_k, a_k, r_k, s_{k+1}) (e.g., it grows as $M \log M$ when coupled with tree-based methods).

- Above a certain number of one-step system transitions, a problem of computational complexity appears.

- In certain situations, one may prefer to rely on algorithms having less inference capabilities than the 'fitted Q iteration algorithm' but which are also less computationally demanding to mitigate this problem of computational complexity (e.g., policy gradient algorithms).

• There is a serious problem plaguing every reinforcement learning algorithm known as the curse of dimensionality[7]: whatever the mechanism behind the generation of the trajectories and without any restrictive assumptions on $f(s, a, w)$, $r(s, a, w)$, S and A, the number of computer operations required to determine (close-to-) optimal policies tends to grow exponentially with the dimensionality of $\mathcal{S} \times \mathcal{A}$.

• This exponentional growth makes these techniques rapidly computationally impractical when the size of the state-action space increases.

• Many researchers in reinforcement learning/dynamic programming/optimal control theory focus their effort on designing algorithms able to break this curse of dimensionality. Deep neural nets give strong hopes for some classes of problems.

[7] A term introduced by Richard Bellman (the founder of the DP theory) in the fifties.

Q-learning with parametric function approximators

Let us extend the Q-learning algorithm to the case where a parametric Q-function of the form $\tilde{Q}_\theta(s, a)$ is used:

1. Equation (39) provides us with a desired update for $\tilde{Q}_\theta(s_t, a_t)$, here: $\delta(s_t, a_t) = r_t + \gamma \max\limits_{a \in \mathcal{A}} \hat{Q}_\theta(s_{t+1}, a) - \hat{Q}_\theta(s_t, a_t)$, after observing (s_t, a_t, r_t, s_{t+1}).

2. It follows the following change in parameters:

$$\theta \leftarrow \theta + \alpha \delta(s_t, a_t) \frac{\partial \tilde{Q}_\theta(s_t, a_t)}{\partial \theta}. \tag{41}$$

Convergence theory of Q-learning

Contraction mapping

Let $B(E)$ be the set of all bounded real-valued functions defined on an arbitrary set E. With every function $R : E \to \mathbb{R}$ that belongs to $B(E)$, we associate the scalar:

$$\|R\|_\infty = \sup_{e \in E} |R(e)|. \tag{42}$$

A mapping $G : B(E) \to B(E)$ is said to be a *contraction mapping* if there exists a scalar $\rho < 1$ such that:

$$\|GR - GR'\|_\infty \leq \rho \|R - R'\|_\infty \quad \forall R, R' \in B(E). \tag{43}$$

Fixed point

$R^* \in B(E)$ is said to be a *fixed point* of a mapping $G : B(E) \to B(E)$ if:

$$GR^* = R^*. \tag{44}$$

If $G : B(E) \to B(E)$ is a *contraction mapping* then there exists a unique fixed point of G. Furthermore if $R \in B(E)$, then

$$\lim_{k\to\infty} \|G^k R - R^*\|_\infty = 0. \tag{45}$$

From now on, we assume that:

1. E is finite and composed of n elements
2. $G : B(E) \to B(E)$ is a contraction mapping whose fixed point is denoted by R^*
3. $R \in B(E)$.

Algorithmic models for computing a fixed point

All elements of R are refreshed: Suppose have the algorithm that updates at stage k $(k \geq 0)$ R as follows:

$$R \leftarrow GR. \tag{46}$$

The value of R computed by this algorithm converges to the fixed point R^* of G. This is an immediate consequence of equation (45).

One element of R is refreshed: Suppose we have the algorithm that selects at each stage k $(k \geq 0)$ an element $e \in E$ and updates $R(e)$ as follows:

$$R(e) \leftarrow (GR)(e) \tag{47}$$

leaving the other components of R unchanged. If each element e of E is selected an infinite number of times then the value of R computed by this algorithm converges to the fixed point R^*.

One element of R is refreshed and noise introduction: Let $\eta \in \mathbb{R}$ be a noise factor and $\alpha \in \mathbb{R}$. Suppose we have the algorithm that selects at stage k ($k \geq 0$) an element $e \in E$ and updates $R(e)$ according to:

$$R(e) \leftarrow (1-\alpha)R(e) + \alpha((GR)(e) + \eta) \tag{48}$$

leaving the other components of R unchanged.

We denote by e_k the element of E selected at stage k, by η_k the noise value at stage k and by R_k the value of R at stage k and by α_k the value of α at stage k. In order to ease further notations we set $\alpha_k(e) = \alpha_k$ if $e = e_k$ and $\alpha_k(e) = 0$ otherwise.

With this notation equation (48) can be rewritten equivalently as follows:

$$R_{k+1}(e_k) = (1-\alpha_k)R_k(e_k) + \alpha_k((GR_k)(e_k) + \eta_k). \tag{49}$$

We define the history $\mathcal{F}_k$ of the algorithm at stage k as being:

$$\mathcal{F}_k = \{R_0, \dots, R_k, e_0, \dots, e_k, \alpha_0, \dots, \alpha_k, \eta_0, \dots, \eta_{k-1}\}. \tag{50}$$

We assume moreover that the following conditions are satisfied:

1. For every k, we have

$$\mathbb{E}[\eta_k | \mathcal{F}_k] = 0. \tag{51}$$

2. There exist two constants A and B such that $\forall k$

$$\mathbb{E}[\eta_k^2 | \mathcal{F}_k] \leq A + B \|R_k\|_\infty^2. \tag{52}$$

3. The $\alpha_k(e)$ are nonnegative and satisfy

$$\sum_{k=0}^{\infty} \alpha_k(e) = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2(e) < \infty. \tag{53}$$

Then the algorithm converges with probability 1 to R^*.

The Q-function as a fixed point of a contraction mapping

We define the mapping $H \colon B(\mathcal{S} \times \mathcal{A}) \to B(\mathcal{S} \times \mathcal{A})$ such that

$$(HK)(s,a) = \underset{w \sim P_w(\cdot)}{\mathbb{E}} [r(s,a,w) + \gamma \max_{a' \in \mathcal{A}} K(f(s,a,w),a')] \tag{54}$$

$\forall (s,a) \in \mathcal{S} \times \mathcal{A}$.

- The recurrence equation (14) for computing the Q_N-functions can be rewritten $Q_N = HQ_{N-1}$ $\forall N > 1$, with $Q_0(s,a) \equiv 0$.

- We prove afterwards that H is a contraction mapping. As immediate consequence, we have, by virtue of the properties algorithmic model (46), that the sequence of Q_N-functions converges to the unique solution of the Bellman equation (18) which can be rewritten: $Q = HQ$. Afterwards, we proof, by using the properties of the algorithmic model (49), the convergence of the Q-learning algorithm.

H is a contraction mapping

This H mapping is a contraction mapping. Indeed, we have for any functions $K, \overline{K} \in B(\mathcal{S} \times \mathcal{A})$:[8]

$$\begin{aligned}
\|HK - H\overline{K}\|_\infty &= \gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big| \mathop{\mathbb{E}}_{w\sim P_w(\cdot)} [\max_{a'\in\mathcal{A}} K(f(s,a,w),a') - \\
&\qquad \max_{a'\in\mathcal{A}} \overline{K}(f(s,a,w),a')]\Big| \\
&\leq \gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big| \mathop{\mathbb{E}}_{w\sim P_w(\cdot)} [\max_{a'\in\mathcal{A}} |K(f(s,a,w),a') - \\
&\qquad \overline{K}(f(s,a,w),a')|]\Big| \\
&\leq \gamma \max_{s\in\mathcal{S}} \max_{a\in\mathcal{A}} |K(s,a) - \overline{K}(s,a)| \\
&= \gamma \|K - \overline{K}\|_\infty
\end{aligned}$$

[8] We make as additional assumption here that the rewards are strictly positive.

Q-learning convergence proof

The Q-learning algorithm updates Q at stage k in the following way[9]

$$\begin{aligned} Q_{k+1}(s_k, a_k) &= (1-\alpha_k)Q_k(s_k, a_k) + \alpha_k(r(s_k, a_k, w_k) + \quad (55) \\ & \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w_k), a)), \quad (56) \end{aligned}$$

Q_k representing the estimate of the Q-function at stage k. w_k is drawn independently according to $P_w(\cdot)$.

[9] The element (s_k, a_k, r_k, s_{k+1}) used to refresh the Q-function at iteration k of the Q-learning algorithm is “replaced” here by $(s_k, a_k, r(s_k, a_k, w_k), f(s_k, a_k, w_k))$.

By using the H mapping definition (equation (54)), equation (56) can be rewritten as follows:

$$Q_{k+1}(s_k, a_k) = (1 - \alpha_k)Q_k(s_k, a_k) + \alpha_k((HQ_k)(s_k, a_k) + \eta_k) \tag{57}$$

with

$$\begin{aligned}\eta_k &= r(s_k, a_k, w_k) + \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w_k), a) - (HQ_k)(s_k, a_k) \\ &= r(s_k, a_k, w_k) + \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w_k), a) - \\ &\quad \mathop{\mathbb{E}}_{w \sim P_w(\cdot|s,a)}[r(s_k, a_k, w) + \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w), a)]\end{aligned}$$

which has exactly the same form as equation (49) (Q_k corresponding to R_k, H to G, (s_k, a_k) to e_k and $\mathcal{S} \times \mathcal{A}$ to E).

We know that H is a contraction mapping. If the $\alpha_k(s_k, a_k)$ terms satisfy expression (53), we still have to verify that η_k satisfies expressions (51) and (52), where

$$\mathcal{F}_k = \{Q_0, \ldots, Q_k, (s_0, a_0), \ldots, (s_k, a_k), \alpha_0, \ldots, \alpha_k, \eta_0, \ldots, \eta_{k-1}\}, \tag{58}$$

in order to ensure the convergence of the Q-learning algorithm.

We have:

$$\begin{aligned} E[\eta_k|\mathcal{F}_k] &= \underset{w_k \sim P_w(\cdot)}{\mathbb{E}}[r(s_k, a_k, w_k) + \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w_k), a) - \\ &\quad \underset{w \sim P_w(\cdot)}{\mathbb{E}}[r(s_k, a_k, w) + \gamma \max_{a \in \mathcal{A}} Q_k(f(s_k, a_k, w), a)]|\mathcal{F}_k] \\ &= 0 \end{aligned}$$

and expression (51) is indeed satisfied.

In order to prove that expression (52) is satisfied, one can first note that :

$$|\eta_k| \leq 2B_r + 2\gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} Q_k(s,a) \tag{59}$$

where B_r is the bound on the rewards. Therefore we have :

$$\eta_k^2 \leq 4B_r^2 + 4\gamma^2 (\max_{(s,a)\in S\times A} Q_k(s,a))^2 + 8B_r\gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} Q_k(s,a) \tag{60}$$

By noting that

$$8B_r\gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} Q_k(s,a) < 8B_r\gamma + 8B_r\gamma (\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} Q_k(s,a))^2 \tag{61}$$

and by choosing $A = 8B_r\gamma + 4B_r^2$ and $B = 8B_r\gamma + 4\gamma^2$ we can write

$$\eta_k^2 \leq A + B\|Q_k\|_\infty^2 \tag{62}$$

and expression (52) is satisfied. **QED**

References

Pascal Leroy, Pablo G. Morato, Jonathan Pisane, Athanasios Kolios, and Damien Ernst. Imp-marl: a suite of environments for large-scale infrastructure management planning via marl, 2023.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Mueller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620:982–987, 08 2023.

Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer Nature, 2022.