Third-year ISAE-SUPAERO engineering students
Research Area: Neuro & AI
December, 2020

Neuro & AI: Methods and Tools for Neuroergonomics

Introduction to Reinforcement Learning

Giorgio Angelotti[1,2] **Nicolas Drougard**[1]

[1]ISAE-SUPAERO DCAS, Toulouse, FRANCE

[2]ANITI, Toulouse, FRANCE

nicolas.drougard@isae-supaero.fr

Outline

References

- [HMS20] G. Haine, D. Matignon, and M. Salaün, *Mathématiques déterministes*, Tronc Commun Scientifique 1A, Formation Ingénieur ISAE-SUPAERO, 2020.
- [MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, *Human-level control through deep reinforcement learning*, Nature **518** (2015), no. 7540, 529–533, download link **here**.
- [Mun13] Rémi Munos, *Lectures: Apprentissage par renforcement*, 2013, download links **here**.
- [SB18] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018, download link **here**.
- [Sil15] David Silver, *Lectures on reinforcement learning*, 2015, download links **here**.
- [Zan05] Elena Zanini, *Markov decision processes*, 2005, download link **here**.

Description of references

- **The** book on Reinforcement Learning (RL) [SB18]
- Lectures on RL by David Silver [Sil15]
- Short presentation of Markov Decision Processes [Zan05]
- French lectures on RL by R.Munos [Mun13]

Outline

Introduction

- RL = Reinforcement Learning
- Learn a behaviour from interaction (with a simulator).
- No supervision (label $y \in \mathcal{Y}$), only a reward signal ($R_t \in \mathbb{R}$).
- Growing field, benefits from Deep Learning.
- Classical RL operates on finite spaces, Deep RL handles continuous ones.
- Videos: Breakout, Locomotion behavior, Alphastar.
- ISAE-SUPAERO RL Initiative SuReLI.

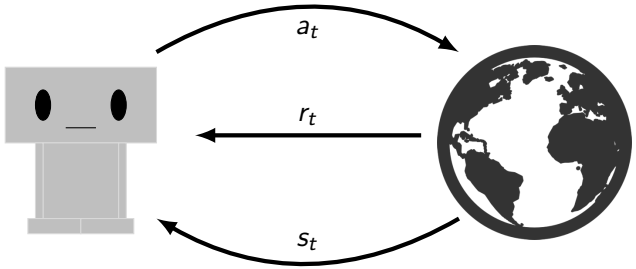
Introduction

Classical Reinforcement learning is based on the **reward hypothesis**.

Reward Hypothesis

A goal can be described by the maximisation of expected cumulative reward $\mathbb{E}\left[\sum_{t\geqslant 0} R_t\right]$.

Do you agree with this statement?



Reinforcement Learning?

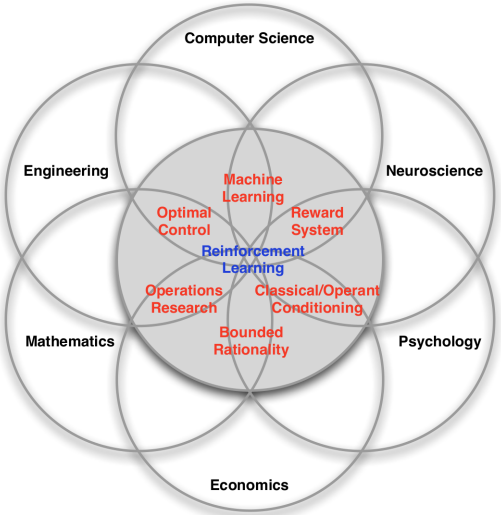


Illustration from the first lecture of David Silver [Sil15].

Outline

The origin of Markov Processes

- Random model set up by Andreï Markov (1856-1922).

- In particular, he studied the letter sequence process in novels.
- Each letter depends primarily on the previous one.

Definition – Markov Chain (or Process)

- $\mathcal{S}$ countable set called *set of states* $s \in \mathcal{S}$.
- $(S_t)_{t\in\mathbb{N}}$ sequence of random variables of $\mathcal{S}$.

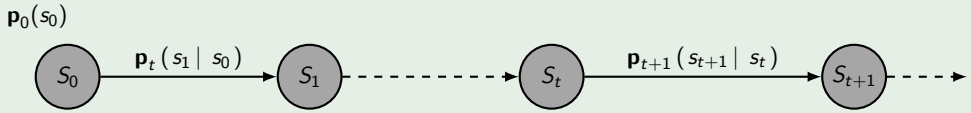
The sequence $(S_t)_{t\in\mathbb{N}}$ is a **Markov Chain** if $\forall t \geqslant 0, \forall (s_0, s_1, \ldots, s_{t+1}) \in \mathcal{S}^{t+2}$

$$\mathbb{P}(S_{t+1} = s_{t+1} \mid S_0 = s_0, S_1 = s_1, \ldots, S_t = s_t) = \mathbb{P}(S_{t+1} = s_{t+1} \mid S_t = s_t).$$

Other equivalent definitions

- "the future is independent of the past given the present"
- $\forall t \geqslant 1$, $S_{t+1} \perp\!\!\!\perp \{S_i\}_{i=0}^{t-1} \mid S_t$.
  note: "$X \perp\!\!\!\perp Y \mid Z$" means "$X$ is independent from Y given Z".
- $S_0 \sim P_0$ and $\forall t > 0$, $S_t \sim P_{s,t}$ if $S_{t-1} = s$,
  where p_0 and $\{P_{s,t}\}_{(s,t)\in\mathcal{S}\times\mathbb{N}}$ are probability distributions on $\mathcal{S}$.
  note: "$X \sim P$" means "X has the probability distribution P".

Graphical representation of a Markov Process



- Graphical representation called **Bayesian Network** (BN).
- In BNs, for each set of nodes N,

$$N \perp\!\!\!\perp \text{non-descendant}(N) \mid \text{parents}(N).$$

- Here, $\text{parents}(S_t) = \{S_{t-1}\}$, $\text{non-descendant}(S_t) = \{S_0, S_1, \dots S_{t-2}\}$.
- For a better readability, $\mathbb{P}(S_{t+1} = s' \mid S_t = s)$ is denoted by $\mathbf{p}_t(s' \mid s)$: $P_{s,t} = \mathbf{p}(. \mid s)$.

Definition – Transition matrix

Let's assume that $\#\mathcal{S} = n$ and let's assign a number to each state $\mathcal{S} = \left\{ s^{(1)}, s^{(2)}, \dots, s^{(n)} \right\}$.

The transition matrix is $\forall (i,j) \in \{1, \dots, n\}^2$, $\forall t \geqslant 0$, $T_{ij}^{(t)} = \mathbf{p}_t \left(s^{(j)} \,\middle|\, s^{(i)} \right)$, *i.e.*

$$T^{(t)} = \begin{pmatrix} \mathbf{p}_t \left(s^{(1)} \,\middle|\, s^{(1)} \right) & \dots & \mathbf{p}_t \left(s^{(n)} \,\middle|\, s^{(1)} \right) \\ \mathbf{p}_t \left(s^{(1)} \,\middle|\, s^{(2)} \right) & \dots & \mathbf{p}_t \left(s^{(n)} \,\middle|\, s^{(2)} \right) \\ \vdots & & \vdots \\ \mathbf{p}_t \left(s^{(1)} \,\middle|\, s^{(n)} \right) & \dots & \mathbf{p}_t \left(s^{(n)} \,\middle|\, s^{(n)} \right) \end{pmatrix} = \begin{pmatrix} P_{s^{(1)},t} \\ P_{s^{(2)},t} \\ \vdots \\ P_{s^{(n)},t} \end{pmatrix}.$$

Distribution of S_t

Consider $p_t \in \mathbb{R}^n$ such that $\forall i \in \{1, \dots, n\}$, $(p_t)_i = \mathbb{P}\left(S_t = s^{(i)} \right)$. Thus, $\forall t \geqslant 0$,

$$\begin{aligned} (p_t)_i &= \sum_{j=1}^{n} \mathbb{P}\left(S_t = s^{(i)}, S_{t-1} = s^{(j)} \right) = \sum_{j=1}^{n} \mathbf{p}_t \left(s^{(i)} \,\middle|\, s^{(j)} \right) \mathbb{P}\left(S_{t-1} = s^{(j)} \right) \\ &= \left(T^{(t)} p_{t-1} \right)_i = \left(T^{(t)} T^{(t-1)} p_{t-2} \right)_i = \dots = \left(T^{(t)} \dots T^{(1)} p_0 \right)_i. \end{aligned}$$

Property of Markov Processes (MPs)

Let $(S_t)_{t\in\mathbb{N}}$ be a Markov Process on space $\mathcal{S}$.
$\forall t \in \mathbb{N}$, $\forall r : \mathcal{S} \to \mathbb{R}$ bounded, $\forall (s_0, \dots, s_t) \in \mathcal{S}^{t+1}$,

$$\begin{aligned}\mathbb{E}\left[r(S_{t+1}) \middle|\ S_0 = s_0, \dots, S_t = s_t\right] &= \sum_{s'\in\mathcal{S}} r(s') \cdot \mathbb{P}\left(S_{t+1} = s' \middle|\ S_t = s_t\right) \\ &= \mathbb{E}\left[r(S_{t+1}) \middle|\ S_t = s_t\right].\end{aligned}$$

Proof

$$\mathbb{E}\left[r(S_{t+1}) \middle|\ S_0 = s_0, \dots, S_t = s_t\right] = \mathbb{E}\left[\sum_{s'\in\mathcal{S}} r(s') \cdot \mathbb{1}_{\{S_{t+1}=s'\}} \middle|\ S_0 = s_0, \dots, S_t = s_t\right]$$

$$\left(\mathbb{1}_{\{e\}} \overset{def}{=} \begin{cases} 1 & \text{if } e \text{ true,} \\ 0 & \text{otherwise.} \end{cases}\right)$$

$$\overset{\text{linear}}{=} \sum_{s'\in\mathcal{S}} r(s') \cdot \mathbb{E}\left[\mathbb{1}_{\{S_{t+1}=s'\}} \middle|\ S_0 = s_0, \dots, S_t = s_t\right]$$

$$\overset{def}{=} \sum_{s'\in\mathcal{S}} r(s') \cdot \mathbb{P}\left(S_{t+1} = s' \middle|\ S_0 = s_0, \dots, S_t = s_t\right)$$

$$\overset{MP}{=} \sum_{s'\in\mathcal{S}} r(s') \cdot \mathbb{P}\left(S_{t+1} = s' \middle|\ S_t = s_t\right) = \mathbb{E}\left[r(S_{t+1}) \middle|\ S_t = s_t\right].$$

Decision Process (DP)

- A DP models a *system* depending on *actions* over *time*.
- Possible *states of the system* are $s \in \mathcal{S}$.
- $\mathbb{N}$ models the *time*, or the *stages of the process*.
- Possible *actions* are denoted by "a" chosen from a finite set $\mathcal{A}$.
- The *agent* = the entity responsible for decision making
  i.e. choosing the successive actions given the current information about the system.
- Example: *DP for a robot*
  - *System state $s \in \mathcal{S}$ features of both the robot and its environment needed to describe the robotic mission, as the robot location or the target location.*
  - *Actions executable by the robot $a \in A$.*
  - *Agent = decision making module of the robot selecting the robot's actions.*

Definition – Markov Decision Process (MDP)

An MDP is a 4-uple $\langle \mathcal{S}, \mathcal{A}, T, r \rangle$, where

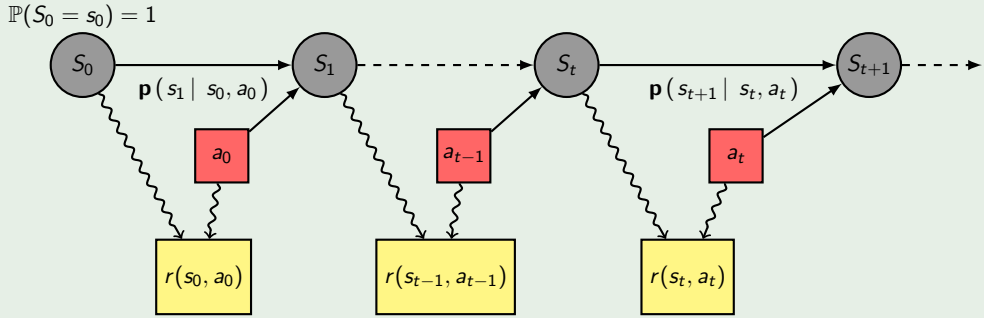
- $\mathcal{S}$ is the finite set of **system states** $s \in \mathcal{S}$,
- $\mathcal{A}$ is the finite set of available **actions** $a \in \mathcal{A}$,
- T is the **transition function**: $T(s, a, s') \stackrel{def}{=} \mathbf{p}(s' \mid s, a)$, $\forall (s, s', a) \in \mathcal{S}^2 \times \mathcal{A}$,
- $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the **reward function**.

.

Remark

If we define a **plan** $(a_t)_{t \in \mathbb{N}}$, *i.e.* a sequence of actions $a_t \in \mathcal{A}$, S_t with transition matrices $T_{ij}^{(t)} = \mathbf{p}\left(s^{(j)} \,\middle|\, s^{(i)}, a_t\right)$ is a Markov Chain.

Graphical representation of a Markov Decision Process



- The gray circles illutrate the Markov process of system states,
- the red squares are decisions, selected actions over time,
- and the yellow squares are collected rewards.

Definition – Strategy

A **strategy** (or policy) is a sequence of functions $\pi = (\pi_t)_{t\in\mathbb{N}}$ such that $\pi_t : \mathcal{S}^{t+1} \to \mathcal{A}$.
Given a strategy π, $\forall$ state **history** $h_t = (s_0, \dots s_t) \in \mathcal{S}^{t+1}$, $\forall t \in \mathbb{N}$, the selected action is

$$a_t = \pi_t(s_0, s_1, \dots, s_t) = \pi_t(h_t).$$

Solving an MDP

- Given a horizon $H \in \mathbb{N}$, the criterion of an MDP is the **Value Function**:

$$V_H^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{H} r\Big(S_t, \pi_t(h_t)\Big) \,\middle|\, S_0 = s\right].$$

- A solution of an MDP is a strategy $(\pi_t^*)_{t\in\mathbb{N}}$ maximizing the value function:

$$\pi^* \in \arg\max_\pi V^\pi(s).$$

Dynamic Programming

To solve the finite horizon problem ($H < +\infty$), let's start from the end:

- Decision step H: what is the optimal choice π_H^*?
  - the agent knows $\{S_0 = s_0, \ldots, S_H = s_H\}$,
  - whatever the previous decisions $(\pi_t)_{t=0}^{H-1}$, π_H^* should maximize
    $a \mapsto \sum_{t=0}^{H-1} r\Big(s_t, \pi_t(s_t)\Big) + r(s_H, a) = cst + r(s_H, a)$.
    $\Rightarrow \pi_H^*$ depends only on s_H: $\pi_H^*(s_H) = \arg\max_{a \in \mathcal{A}} \Big(r(s_H, a)\Big) = \arg\max_{a \in \mathcal{A}} \Big(v_H(s_H, a)\Big)$,
    with $v_H(s, a) = r(s, a)$.

The optimal last decision step π_H^* is known.
Let's define $v_H^*(s) = \max_{a \in \mathcal{A}} v_H(s, a) = \max_{a \in \mathcal{A}} r(s, a)$,
and consider the previous decision step $H - 1$.

Dynamic Programming

- Decision step $H-1$: what is the optimal choice π^*_{H-1}?
  - the agent knows $\{S_0 = s_0, \dots, S_{H-1} = s_{H-1}\}$, and the optimal last decision π^*_H.
  - whatever the previous decisions $(\pi_t)_{t=0}^{H-2}$, π^*_{H-1} should maximize

$$\begin{aligned} a \mapsto & \sum_{t=0}^{H-2} r\Big(s_t, \pi_t(s_t)\Big) + r(s_{H-1}, a) + \mathbb{E}\left[r(S_H, \pi^*_H(S_H)) \mid S_{H-1} = s_{H-1}\right] \\ &= cst + r(s_{H-1}, a) + \mathbb{E}\left[r(S_H, \pi^*_H(S_H)) \mid S_{H-1} = s_{H-1}\right] \\ &= cst + r(s_{H-1}, a) + \sum_{s' \mathcal{S}} T(s_{H-1}, a, s') r\Big(s', \pi^*_H(s')\Big). \\ &= cst + r(s_{H-1}, a) + \sum_{s' \mathcal{S}} T(s_{H-1}, a, s') v^*_H(s'). \end{aligned}$$

  $\Rightarrow$ given π^*_H already computed, π^*_{H-1} depends only on s_{H-1}:

$$\pi^*_{H-1}(s_{H-1}) = \arg\max_{a \in \mathcal{A}} \left(r(s_{H-1}, a) + \sum_{s' \mathcal{S}} T(s_{H-1}, a, s') v^*_H(s') \right) = \arg\max_{a \in \mathcal{A}} \Big(v_{H-1}(s_{H-1}, a) \Big),$$

  where $v^*_H(s) = \max_{a \in \mathcal{A}} v_H(s, a)$ and $v_{H-1}(s, a) = r(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s') v^*_H$.

The optimal two last decisions are known: π^*_H and π^*_{H-1}.

We define $v^*_{H-1}(s) = \max_{a \in \mathcal{A}} v_{H-1}(s, a)$, and consider the previous step $H - n$, with $2 \leqslant n \leqslant H$.

Dynamic Programming

- Decision step $H-n$: what is the optimal choice π^*_{H-n}?
  - the agent knows $\{S_0 = s_0, \ldots, S_{H-n} = s_{H-n}\}$, and optimal decisions from step $H-n+1$.
  - whatever the previous decisions $(\pi_t)_{t=0}^{H-n-1}$, π^*_{H-n} should maximize

$$\begin{aligned} a \mapsto & \sum_{t=0}^{H-n-1} r\Big(s_t, \pi_t(s_t)\Big) + r(s_{H-n}, a) + \mathbb{E}\left[\sum_{t=H-n+1}^{H} r\Big(S_t, \pi^*_t(S_t)\Big) \;\middle|\; S_{H-n} = s_{H-n}\right] \\ &= cst + r(s_{H-n}, a) + \mathbb{E}\left[\sum_{t=H-n+1}^{H} r\Big(S_t, \pi^*_t(S_t)\Big) \;\middle|\; S_{H-n} = s_{H-n}\right] \\ &= cst + r(s_{H-n}, a) + \sum_{s' \mathcal{S}} T(s_{H-n}, a, s') v^*_{H-n+1}(s'). \end{aligned}$$

  $\Rightarrow \pi^*_{H-n}$ depends only on s_{H-n}:

$$\pi^*_{H-n}(s_{H-n}) = \arg\max_{a \in \mathcal{A}} \left(r(s_{H-n}, a) + \sum_{s' \mathcal{S}} T(s_{H-n}, a, s') v^*_{H-n+1}(s') \right),$$

  and we produce the value needed for the previous decision step optimization:

$$v^*_{H-n-1}(s) = \max_{a \in \mathcal{A}} \left(r(s_{H-n}, a) + \sum_{s' \mathcal{S}} T(s_{H-n}, a, s') v^*_{H-n+1}(s') \right).$$

Dynamic Programming

We have just shown by induction that

- π_t^* only depends on s_t
- we can compute an optimal strategy as follows:

$$\begin{aligned}
\pi_H^*(s) &= \arg\max_{a \in \mathcal{A}} \Big(r(s,a) \Big), \\
v_H^*(s) &= \max_{a \in \mathcal{A}} \Big(r(s,a) \Big),
\end{aligned}$$

and for t from $H-1$ to 0:

$$\begin{aligned}
\pi_t^*(s) &= \arg\max_{a \in \mathcal{A}} \left(r(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') v_{t+1}^*(s') \right), \\
v_t^*(s) &= \max_{a \in \mathcal{A}} \left(r(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') v_{t+1}^*(s') \right).
\end{aligned}$$

Consequences on strategies

A strategy can then be defined as

$$\pi_t : \mathcal{S} \to \mathcal{A}.$$

Note: in the litterature, it can also return a probability distribution on $\mathcal{A}$.

Consequence on MDPs

Given a strategy π, $(S_t)_{t\in\mathbb{N}}$ is a Markov Chain with transition matrices $T_{ij}^{(t)} = \mathbf{p}\left(s^{(j)} \,\middle|\, s^{(i)}, \pi_t(s^{(i)})\right)$.

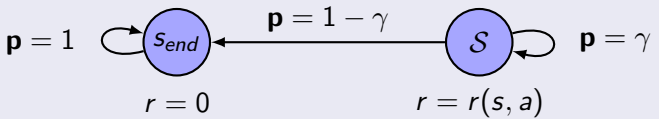
Introduction to the infinite horizon case ($H = +\infty$)

Issue with case $H = +\infty$: $\sum_{t=0}^{+\infty} r(s_t, a_t)$ not always finite.

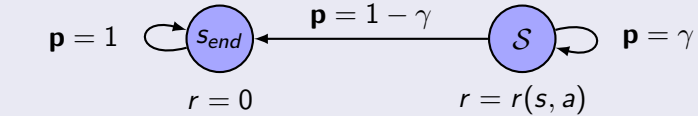
Overcoming this issue by defining a new MDP $\widetilde{\mathcal{M}}$ with an extra state s_{end}.

- s_{end} absorbing state (or dead end): $\mathbf{p}(s_{end} \mid s_{end}, a) = 1$, $\forall a \in \mathcal{A}$.
- $r(s_{end}, a) = 0$, $\forall a \in \mathcal{A}$.
- $\mathbf{p}(s_{end} \mid s, a) = 1 - \gamma \in]0, 1[$, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$.

$\widetilde{\mathcal{M}}$:



Introduction to the infinite horizon case ($H = +\infty$)



From the initial MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, r \rangle$, a new MDP $\widetilde{\mathcal{M}} = \langle \widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{T}, \widetilde{r} \rangle$ is defined, with:

- a system space $\widetilde{\mathcal{S}} = \mathcal{S} \cup \{ s_{end} \}$.
- a new transition function $\widetilde{T}(s, a, s') = \begin{cases} \gamma T(s, a, s') & \text{if} \quad (s, s') \in \mathcal{S}^2 \\ (1 - \gamma) & \text{if} \quad s \in \mathcal{S}, s' = s_{end}, \\ 1 & \text{if} \quad s = s' = s_{end}, \\ 0 & \text{if} \quad s = s_{end}, s' \in \mathcal{S}. \end{cases}$
- a new reward function $\widetilde{r}(s, a) = \begin{cases} r(s, a) & \text{if} \quad s \in \mathcal{S}, \\ 0 & \text{if} \quad s = s_{end}. \end{cases}$

Value function of the new MDP $\widetilde{\mathcal{M}}$

If $S_0 = s_{end}$, $S_t = s_{end}\ \forall t \geqslant 0$: $\widetilde{\mathcal{M}}$'s value function is $\widetilde{V}^{\pi}_{H}(s_{end}) = \sum_{t=0}^{H} r\Big(s_{end}, \pi_t(h_t)\Big) = 0$.

Consider now $s \in \mathcal{S}$, *i.e.* $s \neq s_{end}$:

$$
\begin{aligned}
\widetilde{V}^{\pi}_{0}(s) &= \mathbb{E}_{S_t \sim \widetilde{T}}\Big[\, r\Big(S_0, \pi_0(S_0)\Big) \,\Big|\, S_0 = s \Big] = r\Big(s, \pi_0(s)\Big). \\
\widetilde{V}^{\pi}_{1}(s) - \widetilde{V}^{\pi}_{0}(s) &= \mathbb{E}_{S_t \sim \widetilde{T}}\Big[\, r\Big(S_1, \pi_1(S_1)\Big) \,\Big|\, S_0 = s \Big] = \sum_{s' \in \widetilde{\mathcal{S}}} \widetilde{T}\Big(s, \pi_1(s'), s'\Big) r\Big(s', \pi_1(s')\Big) \\
&= (1-\gamma) r\Big(s_{end}, \pi_1(s_{end})\Big) + \sum_{s' \in \mathcal{S}} \gamma T\Big(s, \pi_1(s'), s'\Big) r\Big(s', \pi_1(s')\Big) \\
&= 0 + \gamma \sum_{s' \in \mathcal{S}} T\Big(s, \pi_1(s'), s'\Big) r\Big(s', \pi_1(s')\Big) \\
&= \mathbb{E}_{S_t \sim T}\Big[\, \gamma r\Big(S_1, \pi_1(S_1)\Big) \,\Big|\, S_0 = s \Big].
\end{aligned}
$$

Value function of the new MDP $\widetilde{\mathcal{M}}$

Using strategy $(\pi_t)_{t\in N}$, and considering $f : \mathcal{S} \to \mathbb{R}$ with $f(s_{end}) = 0$:

$$\begin{aligned}
\mathbb{E}_{S_{t+1}\sim\widetilde{T}}\left[f\Big(S_{t+1}\Big) \,\middle|\, S_t = s \right] &= \sum_{s'\in\widetilde{\mathcal{S}}} \widetilde{T}\Big(s, \pi_t(s), s'\Big) f(s') \\
&= (1-\gamma) f(s_{end}) + \sum_{s'\in\mathcal{S}} \gamma T\Big(s, \pi_1(s'), s'\Big) f(s') \\
&= 0 + \gamma \sum_{s'\in\mathcal{S}} T\Big(s, \pi_1(s'), s'\Big) f(s') \\
&= \gamma \mathbb{E}_{S_{t+1}\sim T}\left[f\Big(S_{t+1}\Big) \,\middle|\, S_t = s \right].
\end{aligned}$$

Some properties of the conditional expectation

$X \in \mathcal{X}$, $Y \in \mathcal{Y}$ random variables, $f : \mathcal{X} \to \mathbb{R}$ a function of X returning real numbers.

- $\mathbb{E}[f(X) \mid Y]$ is a function of Y:

$$\sum_{x \in \mathcal{X}} f(x) \mathbb{P}(X = x \mid Y).$$

- $\mathbb{E}[f(X) \mid X] = f(X)$.
- $\mathbb{E}\Big[\mathbb{E}[f(X) \mid Y] \Big| Z\Big] = \mathbb{E}[f(X) \mid Z]$.

Value function of the new MDP $\widetilde{\mathcal{M}}$

$$
\begin{aligned}
\widetilde{V}_H^{\pi}(s) - \widetilde{V}_{H-1}^{\pi}(s) &= \mathbb{E}_{S_t \sim \widetilde{T}}\left[r\Big(S_H, \pi_H(S_H)\Big) \,\middle|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[f(S_H) \,|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\mathbb{E}_{S_t \sim \widetilde{T}}\left[f(S_H) \,|\, S_{H-1} \right] \,\middle|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma \mathbb{E}_{S_t \sim T}\left[f(S_H) \,|\, S_{H-1} \right] \,|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma g(S_{H-1}) \,|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma \mathbb{E}_{S_t \sim \widetilde{T}}\left[g(S_{H-1}) \,|\, S_{H-2} \right] \,\middle|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma^2 \mathbb{E}_{S_t \sim T}\left[g(S_{H-1}) \,|\, S_{H-2} \right] \,\middle|\, S_0 = s \right] \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma^2 \mathbb{E}_{S_t \sim T}\left[f(S_H) \,|\, S_{H-2} \right] \,\middle|\, S_0 = s \right] \\
&= \dots \\
&= \mathbb{E}_{S_t \sim \widetilde{T}}\left[\gamma^H \mathbb{E}_{S_t \sim T}\left[f(S_H) \,|\, S_0 \right] \,\middle|\, S_0 = s \right] = \gamma^H \mathbb{E}_{S_t \sim T}\left[f(S_H) \,|\, S_0 = s \right].
\end{aligned}
$$

Value function of the new MDP $\widetilde{\mathcal{M}}$

$$\begin{aligned}\widetilde{V}_H^\pi(s) = \sum_{t=0}^{H} \widetilde{V}_t^\pi(s) - \widetilde{V}_{t-1}^\pi(s) &= \sum_{t=0}^{H} \gamma^t \mathbb{E}_{S_t \sim T}\left[r\Big(S_t, \pi_t(S_t)\Big) \,\middle|\, S_0 = s \right] \\ &= \mathbb{E}_{S_t \sim T}\left[\sum_{t=0}^{H} \gamma^t r\Big(S_t, \pi_t(S_t)\Big) \,\middle|\, S_0 = s \right].\end{aligned}$$

Since $\displaystyle\sum_{t \geqslant 0} \gamma^t = \frac{1}{1-\gamma}$, $\displaystyle\sum_{t=0}^{H} \gamma^t r\Big(S_t, \pi_t(S_t)\Big) \leqslant \frac{1}{1-\gamma} \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} r(s,a)$,
and the value function of $\widetilde{\mathcal{M}}$ can be computed for $H \to +\infty$.

Value function for an infinite horizon

$$V^\pi(s) = \mathbb{E}\left[\sum_{t \geqslant 0} \gamma^t r\Big(S_t, \pi_t(S_t)\Big) \,\middle|\, S_0 = s \right], \text{ with } 0 < \gamma < 1 \text{ (discount factor).}$$

Bellman Equation – Let $\pi_{t\in\mathbb{N}}$ be a strategy.

$$
\begin{aligned}
V^{\pi}(s) &:= \mathbb{E}\left[\sum_{t=0}^{+\infty}\gamma^t\cdot r\Big(S_t,\pi_t(S_t)\Big)\,\middle|\, S_0=s\right]\\
&= \mathbb{E}\left[r\Big(S_0,\pi_0(S_0)\Big)+\sum_{t=1}^{+\infty}\gamma^t\cdot r\Big(S_t,\pi_t(S_t)\Big)\,\middle|\, S_0=s\right]\\
&= r\Big(s,\pi_0(s)\Big)+\mathbb{E}\left[\mathbb{E}\Big[\sum_{t=1}^{+\infty}\gamma^t\cdot r\Big(S_t,\pi_t(S_t)\Big)\Big|S_1\Big]\,\middle|\, S_0=s\right]\\
&= r\Big(s,\pi_0(s)\Big)+\gamma\cdot\sum_{s'\in\mathcal{S}}\mathbf{p}\Big(s'\Big|s,\pi_0(s)\Big)\cdot\mathbb{E}\left[\sum_{t=1}^{+\infty}\gamma^{t-1}\cdot r\Big(S_t,\pi_t(S_t)\Big)\,\middle|\, S_1=s'\right]\\
&= r\Big(s,\pi_0(s)\Big)+\gamma\cdot\sum_{s'\in\mathcal{S}}\mathbf{p}\Big(s'\Big|s,\pi_0(s)\Big)\cdot\mathbb{E}\left[\sum_{t'=0}^{+\infty}\gamma^{t'}\cdot r\Big(S^+_{t'},\pi^+_{t'}(S^+_{t'})\Big)\,\middle|\, S^+_0=s'\right]\\
&= r\Big(s,\pi_0(s)\Big)+\gamma\cdot\sum_{s'\in\mathcal{S}}\mathbf{p}\Big(s'\Big|s,\pi_0(s)\Big)\cdot V^{\pi^+}(s'),\ \text{with } S^+_t=S_{t+1},\ \pi^+_t(s)=\pi_{t+1}(s).
\end{aligned}
$$

Bellman Equation

For a strategy $(\pi_t)_{t\geqslant 0}$,

$$V^{\pi}(s) = r\Big(s, \pi_0(s)\Big) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbf{p}\Big(s' \Big| s, \pi_0(s)\Big) \cdot V^{\pi^+}(s'),$$

with $\forall s \in \mathcal{S}$, $\forall t \geqslant 0$, $\pi_t^+(s) = \pi_{t+1}(s)$.

Dynamic Programming Equation

The optimal value function $V^* = V^{\pi^*}$ is the unique solution of the following equation:

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

There exists an optimal strategy $\pi^* : \mathcal{S} \to \mathcal{A}$, independent from the time step $t \in \mathbb{N}$, that is $\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbf{p}(s'|s,a) \cdot V^*(s') \right)$.

Q-value

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{p}\left(s' \mid s,a\right) V^{\pi}(s').$$

Optimal Q-value

$$Q^{*}(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{p}\left(s' \mid s,a\right) V^{*}(s').$$

$\Rightarrow \pi^{*}(s) = \arg\max_{a \in \mathcal{A}} Q^{*}(s,a).$

Value Iteration Algorithm

Initialize: $Q = r$

Repeat:

- $V(s) = \max_{a \in \mathcal{A}} Q(s, a)$
- $Q(s) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{p}(s' \mid s, a)\, V(s')$

(*i.e.* DP equations) until convergence
(thanks to the Fixed Point/Contraction Mapping Theorem (Thm 7.1 [HMS20]):
Bellman & DP operators are contracting).

Return:

- $\pi^*(s) = \max_a Q(s, a)$.

With matrices

Denote states by $\mathcal{S} = \left\{ s^{(1)}, s^{(2)}, \ldots, s^{(n)} \right\}$, with $n = \#\mathcal{S}$.

Define

- action vector $a \in \mathcal{A}^n$,
- strategy vector $\pi \in \mathcal{A}^n$: $\pi_i = \pi(s^{(i)}), \qquad \forall i \in \{1, \ldots, n\}$,
- reward vector $r^a \in \mathbb{R}^n$: $r^a_i = r(a_i, s^{(i)}), \qquad \forall i \in \{1, \ldots, n\}$,
- transition matrix $T^a \in \mathbb{R}^{n \times n}$: $T^a_{ij} = \mathbf{p}\left(s^{(j)} \,\middle|\, s^{(i)}, a_i \right), \qquad \forall (i,j) \in \{1, \ldots, n\}^2$,
- value vector $V \in \mathbb{R}^n$: $V_i = V(s^{(i)}), \qquad \forall i \in \{1, \ldots, n\}$,

The Bellman Equation becomes

$$V^\pi = r^\pi + \gamma T^\pi V^\pi \Rightarrow (I_n - \gamma T^\pi) V^\pi = r^\pi.$$

The Dynamic Programmation Equation becomes

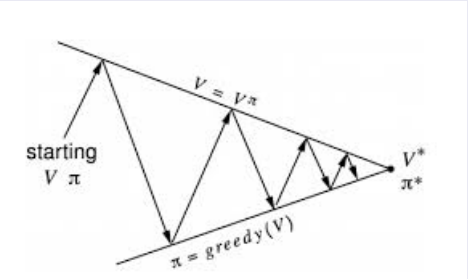
$$V^* = \max_{a \in \mathcal{A}^n} r^a + \gamma T^a V^*.$$

Policy Iteration Algorithm

Initialize $Q = r$, $\pi = a \in \mathcal{A}^n$, and repeat:

- $\pi \in \arg\max_{a \in \mathcal{A}^n} Q$,
- resolution of $(I_n - \gamma T^\pi)V = r^\pi$ (or iterations of Bellman equation)
- $Q = r^\pi + \gamma T^\pi V$.

until convergence (*i.e.* when π remains the same).



Policy Iteration Algorithm always converges to π^*

Given π, $\forall s \in \mathcal{S}$, $\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$ is a better action: $V^{\pi}(s) = Q^{\pi}\Big(s, \pi(s)\Big)$

$$
\begin{aligned}
\leqslant \max_{a \in \mathcal{A}} Q^{\pi}(s,a) &= Q^{\pi}\Big(s, \pi'(s)\Big) \\
&= r\Big(s, \pi'(s)\Big) + \gamma \sum_{s' \in \mathcal{S}} T\Big(s, \pi'(s), s'\Big) V^{\pi}(s') \\
&= r\Big(s, \pi'(s)\Big) + \gamma \sum_{s' \in \mathcal{S}} T\Big(s, \pi'(s), s'\Big) Q^{\pi}\Big(s', \pi(s')\Big) \\
&\leqslant r\Big(s, \pi'(s)\Big) + \gamma \sum_{s' \in \mathcal{S}} T\Big(s, \pi'(s), s'\Big) Q^{\pi}\Big(s', \pi'(s')\Big) \\
&\leqslant r\Big(s, \pi'(s)\Big) + \gamma \sum_{s' \in \mathcal{S}} T\Big(s, \pi'(s), s'\Big) \left[r\Big(s', \pi'(s)\Big) + \gamma \sum_{s'' \in \mathcal{S}} \dots \right] \\
&\leqslant \dots \\
&\leqslant V^{\pi'}(s).
\end{aligned}
$$

Policy Iteration Algorithm always converges to π^*

If improvement stops, *i.e.* $\max_{a \in \mathcal{A}} Q^{\pi}(s,a) = Q^{\pi}\Big(s, \pi(s)\Big) = V^{\pi}(s)$, $\forall s \in \mathcal{S}$. Thus V^{π} satisfies the DP equation $V^{\pi} = \max_{a \in \mathcal{A}} Q^{\pi}(s,a)$, and then

$$V^{\pi} = V^{*}.$$

That means that π is optimal!

Learning and Planning

Two fundamental problems in sequential decision making under uncertainty:

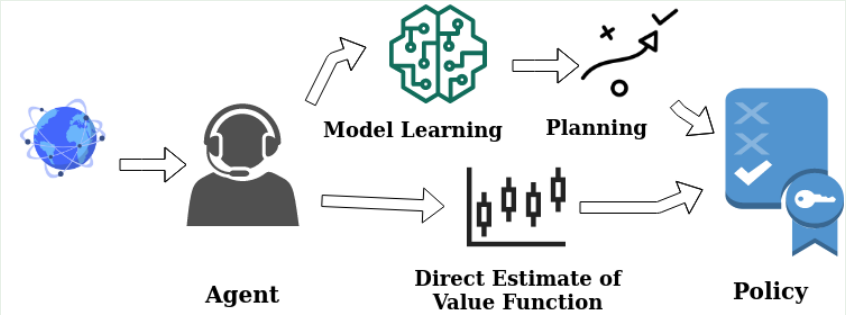
- Reinforcement Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
- Planning:
  - A model of the environment is known ($\mathcal{M}$)
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy

Outline

Model based/Model free Reinforcement Learning

In RL, the MDP $\mathcal{M}$ is unknown. However the agent can interact with the associated environment, and compute some estimates to improve the strategy.

- model based: estimation of T and optimization of the resulting MDP.
- model free: estimation of Q^*, or directly π^*.



Vocabulary: different types of RL

- Value Based
  - No Policy (Implicit, $\arg\max_{a \in \mathcal{A}} Q(s, a)$)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

A first learning method: averaging sample returns,
a.k.a. **Monte Carlo** method.

Monte Carlo Policy Evaluation

- Consider a trajectory using strategy π: $(S_0, S_1, \ldots, S_T) \sim T^\pi$,
- Define the **return** of the state S_t:

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r\Big(S_{t'}, \pi(S_{t'})\Big) = R_t + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots + \gamma^{T-t} R_T,$$

  with $R_t = r\Big(S_t, \pi(S_t)\Big)$.
- Since γ^t goes to 0 exponentially, V^π is close to the expected return:

$$V^\pi(s) \approx \mathbb{E}\left[\, G_t \,|\, S_t = s \right].$$

- Monte-Carlo policy evaluation uses empirical mean return instead of expected return

Remark

The mean μ_k of a sequence $(x_i)_{i=1}^k$ can be computed incrementaly:

$$\begin{aligned}
\mu_{k+1} &= \frac{1}{k+1}\sum_{i=1}^{k+1} x_i = \frac{x_{k+1}}{k+1} + \frac{1}{k+1}\sum_{i=1}^{k} x_i \\
&= \frac{x_{k+1}}{k+1} + \frac{k}{k+1}\mu_k \\
&= \frac{1}{k+1}\left(x_{k+1} + k\mu_k\right) \\
&= \frac{1}{k+1}\Big(x_{k+1} + (k+1)\mu_k - \mu_k\Big) \\
&= \mu_k + \frac{1}{k+1}\left(x_{k+1} - \mu_k\right).
\end{aligned}$$

First-visit Monte-Carlo prediction: estimation of V^π

input : strategy π, number of episodes n
output: V, estimation of V^π
$N(s) \leftarrow 0, \forall s \in \mathcal{S}$;
for i **in** $\{1, \ldots, n\}$ **do**
  generate $(s_0, s_1, \ldots, s_T)$ using π;
  visited $\leftarrow \varnothing$;
  for t **in** $\{0, \ldots, T\}$ **do**
    if $s_t \notin$ *visited* **then**
      $N(s_t) \leftarrow N(s_t) + 1$;
      $G(s_t) \leftarrow \sum_{i=t}^{T} \gamma^{i-t} r\Big(s_i, \pi(s_i)\Big)$;
      $V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}\Big(G(s_t) - V(s_t)\Big)$;
      visited.add(s_t);

Remarks

- $V(s)$ is the mean of independent identically distributed (i.i.d.) samples G.
- Strong Law of Large Numbers (SLLN) $\Longrightarrow V(s) \xrightarrow[n\to+\infty]{a.s.} V^\pi$.

$$\begin{aligned}
\mathbb{E}\left[V(s)\right] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} G_i\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[G_i\right] = V^\pi(s) \\
Var\Big(V(s)\Big) &= Var\left(\frac{1}{n}\sum_{i=1}^{n} G_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var\left(G_i\right) = \frac{Var(G)}{n} \\
\mathbb{E}\left[\Big(V(s) - V^\pi(s)\Big)^2\right] &= \mathbb{E}\left[V(s) - \mathbb{E}\left[V(s)\right]\right]^2 + Var\Big(V(s)\Big) = 0 + \frac{Var(G)}{n} \sim \frac{1}{n}.
\end{aligned}$$

- "Every visit MC" consists in removing the condition "$s_t \notin$ visited".
- In this case, the samples G are not independent anymore, but it converges too.
- Q^π can be estimated in the same way.

Introduction to Temporal Differences (TD)

First, consider again the estimation update: $V(s) \leftarrow V(s) + \frac{1}{N(s)}\Big(G(s) - V(s)\Big)$.

Fixing $N \Rightarrow$ old episodes forgotten with updates.

With $\forall s \in \mathcal{S}$, $N(s) = \frac{1}{\alpha} \in]0, 1[$, we get: $V(s) \leftarrow V(s) + \alpha\Big(G(s) - V(s)\Big)$.

Another idea is to use $R_t + \gamma V(S_{t+1})$ instead of $G(S_t)$:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big(R_t + \gamma V(S_{t+1}) - V(S_t)\Big).$$

TD vocabulary

- $R_t + \gamma V(S_{t+1})$ is called the **TD target**,
- $R_t + \gamma V(S_{t+1}) - V(S_t)$ the **TD error**.

- TD learns online after every step
  $\rightarrow$ learns from incomplete sequences.
- MC must wait until end of episode before return is known
  $\rightarrow$ learns from complete sequences.
- TD works in continuing (non-terminating) environments
- MC only works for episodic (terminating) environments

Let's consider a batch of K episodes:

$$
\begin{aligned}
e_1 &= s_0^{(1)}, a_0^{(1)}, r_0^{(1)}, \quad s_1^{(1)}, a_1^{(1)}, r_1^{(1)}, \quad \dots, \; s_{H_1}^{(1)} \\
e_2 &= s_0^{(2)}, a_0^{(2)}, r_0^{(2)}, \quad s_1^{(2)}, a_1^{(2)}, r_1^{(1)}, \quad \dots, \; s_{H_2}^{(2)} \\
&\vdots \\
e_K &= s_0^{(K)}, a_0^{(K)}, r_0^{(K)}, \quad s_1^{(K)}, a_1^{(K)}, r_1^{(K)}, \quad \dots, \; s_{H_K}^{(K)}
\end{aligned}
$$

Monte-Carlo:

- $K(s) = \sum_{k=1}^{K} \mathbb{1}_{\{ s \in e_k \}} \leqslant K$, $t_k(s) = \min \{ t \in \{0, \dots, H_k\} \mid s_t \in e_k \text{ and } s_t = s \}$.
- $\forall s \in \mathcal{S}$ such that $K(s) > 0$, $V(s) = \dfrac{1}{K(s)} \displaystyle\sum_{\substack{k \text{ s.t.} \\ s \in e_k}} \sum_{t=t_k(s)}^{H_k - 1} \gamma^{t - t_k(s)} r_t^{(k)} = \dfrac{1}{K(s)} \displaystyle\sum_{\substack{k \text{ s.t.} \\ s \in e_k}} g^{(k)}(s)$.
- $V(s)$ minimizes the least squares $\displaystyle\sum_{\substack{k \text{ s.t.} \\ s \in e_k}} \left(V(s) - g^{(k)}(s) \right)^2$.

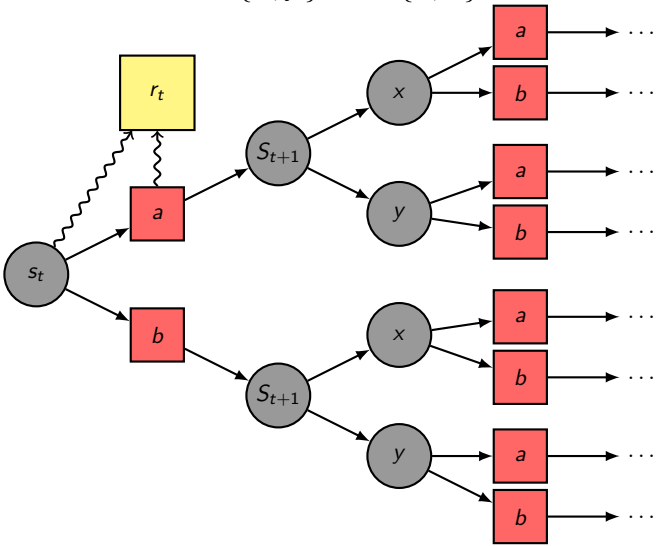
Let's consider a batch of K episodes:

$$
\begin{aligned}
e_1 &= s_0^{(1)}, a_0^{(1)}, r_0^{(1)}, \quad s_1^{(1)}, a_1^{(1)}, r_1^{(1)}, \quad \ldots, \; s_{H_1}^{(1)} \\
e_2 &= s_0^{(2)}, a_0^{(2)}, r_0^{(2)}, \quad s_1^{(2)}, a_1^{(2)}, r_1^{(1)}, \quad \ldots, \; s_{H_2}^{(2)} \\
&\vdots \\
e_K &= s_0^{(K)}, a_0^{(K)}, r_0^{(K)}, \quad s_1^{(K)}, a_1^{(K)}, r_1^{(K)}, \quad \ldots, \; s_{H_K}^{(K)}
\end{aligned}
$$

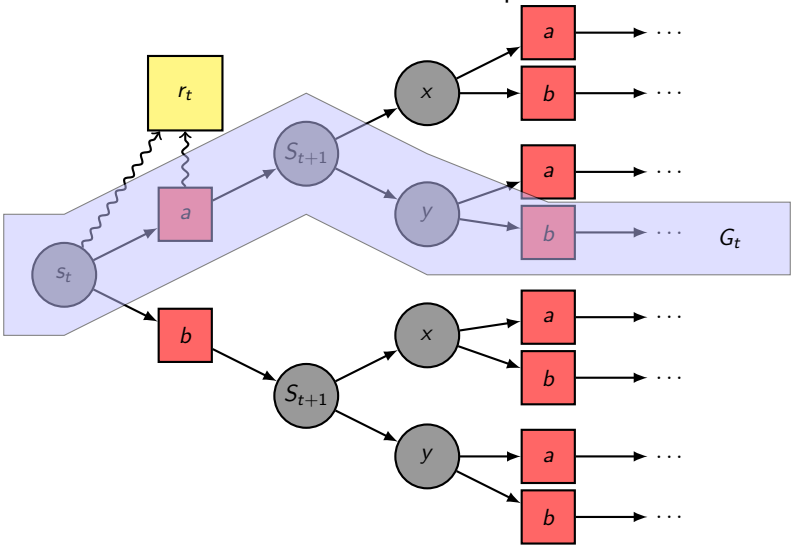
Temporal Differences TD(0):

- Under some conditions, TD(0) converges to solution of max likelihood Markov model.
- Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \widehat{T}, \widehat{r}, \gamma \rangle$ with
  - $\widehat{T}(s,a,s') = \dfrac{1}{N(s,a)} \displaystyle\sum_{k=1}^{K} \sum_{t=0}^{H_k-1} \mathbb{1}_{\left\{ (s,a,s') = (s_t^{(k)}, a_t^{(k)}, s_{t+1}^{(k)}) \right\}}$, and
  - $\widehat{R}(s,a) = \dfrac{1}{N(s,a)} \displaystyle\sum_{k=1}^{K} \sum_{t=0}^{H_k-1} r_t^{(k)} \mathbb{1}_{\left\{ (s,a) = (s_t^{(k)}, a_t^{(k)}) \right\}}$.

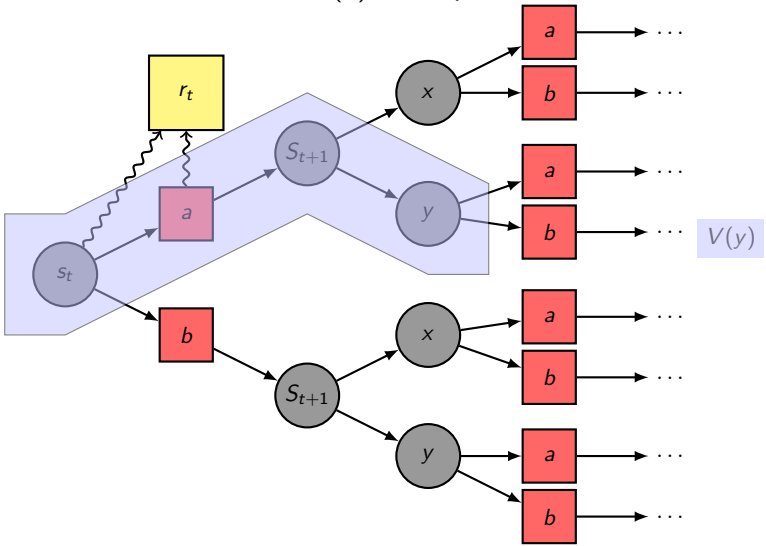
$\mathcal{S} = \{x, y\}$, $\mathcal{A} = \{a, b\}$:



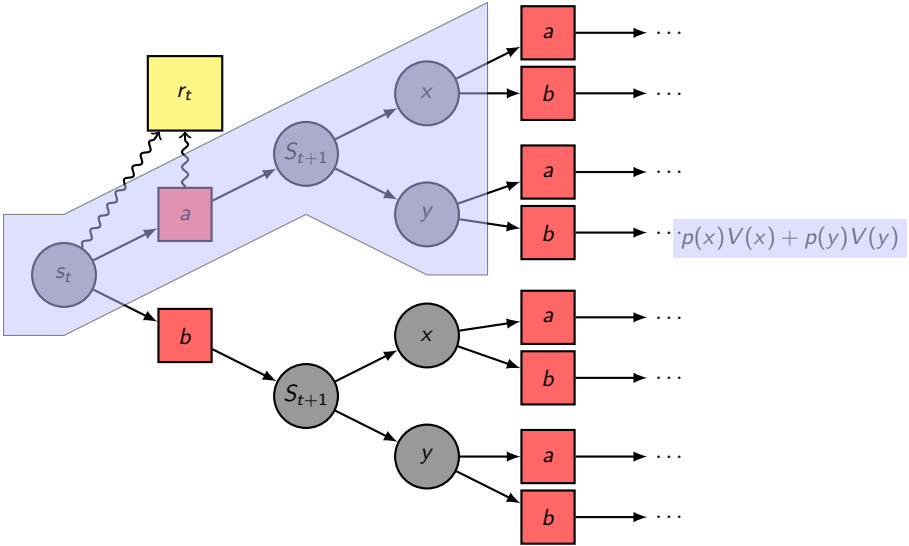
Monte Carlo Backup:



TD(0) Backup:



Dynamic Programming Backup:



Bootstrap & Sampling

- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

TD(λ) with $\lambda \in [0,1]$

- n-step return: $G_t^n = R_t + \gamma R_{t+1} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$
  - G_t^∞ is the classical return $G_t = \sum_{t'=t}^{H-1} \gamma^{t'-t} R_{t'}$
  - G_t^1 is the TD target $G_t^0 = R_t + \gamma V(S_{t+1})$.
- λ-return $G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^n$

  average of $\left(G_t^n\right)_{n \geqslant 1}$ weighted by a geometric distribution with parameter $1-\lambda$.
- TD(λ) Backup:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big(G_t^\lambda - V(S_t)\Big).$$

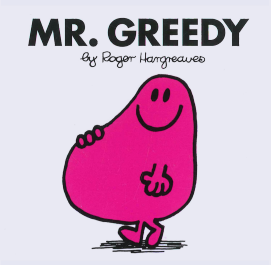
Greedy policy ?

- Greedy policy improvement over value function V **requires MDP model**:

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V(s') \right\}.$$

- Greedy policy improvement over Q-value function Q is **model free**:

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



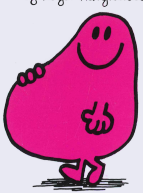
Greedy policy improvement

Let $\pi : \mathcal{S} \to \mathcal{A}$ be a strategy. Consider the greedy policy π' with respect to Q^π:

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

Then $V^{\pi'} \geqslant V^\pi$ (see slide 35).

Epsilon-greedy policy ?

$$(1-\varepsilon) \times$$ 

$$+ \quad \varepsilon \times$$ 

- For a **random policy** π, $\forall s \in \mathcal{S}$, $\pi(s)$ is a probability distribution over actions $\mathcal{A}$, with values denoted by $\pi(a \mid s) \in [0,1]$, $\forall a \in \mathcal{A}$.
- An ε-**greedy** policy π is the following random policy:

$$\pi(a \mid s) = \begin{cases} 1-\varepsilon+\frac{\varepsilon}{\#\mathcal{A}} & \text{if } a = a^*, \\ \frac{\varepsilon}{\#\mathcal{A}} & \text{otherwise.} \end{cases}, \text{ with } a^* \in \arg\max_{a \in \mathcal{A}} Q(s,a) \text{ (greedy action).}$$

Epsilon-greedy policy improvement

Let $\pi : \mathcal{S} \to \mathcal{A}$ be an ε-greedy strategy. Consider the ε-greedy policy π' with respect to Q^π:

$$\pi'(a \mid s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{\#\mathcal{A}} & \text{if } a = a^*, \\ \frac{\varepsilon}{\#\mathcal{A}} & \text{otherwise.} \end{cases} \quad \text{with } a^* \in \arg\max_{a \in \mathcal{A}} Q^\pi(s, a) \ .$$

Note that $\frac{\varepsilon}{\#\mathcal{A}} \leqslant \pi(a \mid s) \leqslant 1 - \varepsilon + \frac{\varepsilon}{\#\mathcal{A}}$. We have:

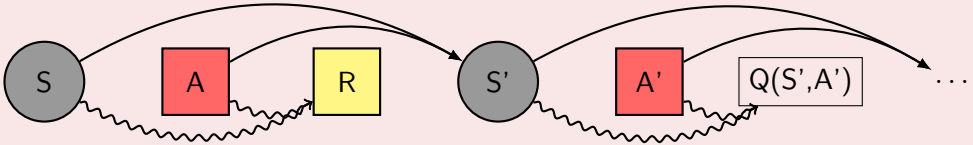
$$\begin{aligned} V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s)\, Q^\pi(s, a) &= (1 - \varepsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a \mid s) - \frac{\varepsilon}{\#\mathcal{A}}}{1 - \varepsilon} Q^\pi(s, a) + \frac{\varepsilon}{\#\mathcal{A}} \sum_{a \in \mathcal{A}} Q^\pi(s, a) \\ &\leqslant (1 - \varepsilon) \max_{a \in \mathcal{A}} Q^\pi(s, a) + \frac{\varepsilon}{\#\mathcal{A}} \sum_{a \in \mathcal{A}} Q^\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi'(a \mid s)\, Q^\pi(s, a) = Q^\pi\Big(s, \pi'(s)\Big) \leqslant V^{\pi'}(s), \end{aligned}$$

with last inequality from slide 35.

Learn and act with ε-greedy policies

- ε-greedy policies can be used with MC and TD(λ) for control
- $(1-\varepsilon)$ is the proportion of **exploitation**, ε of **exploration**
- another algorithm based on the estimation of the Q-value function is called SARSA.

SARSA



$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s,a)\Big).$$

SARSA

input : arbitrarily initialized Q-value
output: improved Q
for i **in** $\{1, \ldots, n\}$ **do**
    initialize $s \sim p_{s_0}$;
    $a \leftarrow epsilon\_greedy(Q, s)$;
    while *episode not finished* **do**
        $(r, s') \leftarrow next\_state(s, a)$;
        $a' \leftarrow epsilon\_greedy(Q, s')$;
        $Q(s, a) \leftarrow Q(s, a) + \alpha\Big(r + \gamma Q(s', a') - Q(s, a)\Big)$;
        $a \leftarrow a'$;
        $s \leftarrow s'$;

Remarks

- SARSA(λ) is the version with the averaged n-step returns
  cf. slide 54 for TD(λ).
- SARSA is said **on policy** because the same policy is used for control and Q-update.

What if π does not accomplish to remove the condition $\exists s : s \notin \textit{visited}$?
Then we can not fix N to forget old episodes, because some $N(s) = 0 \implies \alpha = \frac{1}{N} \to +\infty$.
Idea: change π while collecting $(s_0, s_1 \ldots, s_T)$ to ensure that all "relevant" s are explored.
Set the learning rate $0 < \alpha < 1$. At each step t

Q-learning

- $\varepsilon-$greedy exploration strategy.
  Select the action:
  - random action $a \in \mathcal{A}$ with probability ε (exploration),
  - $a_t \in \arg\max_{a \in \mathcal{A}} Q(s_{t+1}, a)$ with probability $1 - \varepsilon$ (exploitation).
- Receive the state the reward $r(s_t, a_t)$ and next state s_{t+1}.
- Update the Q-value estimate:

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot \Big(r(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a) \Big).$$

Q-learning

input : arbitrarily initialized Q-value
output: improved Q
for i **in** $\{1, \ldots, n\}$ **do**
    initialize $s \sim p_{s_0}$;
    $a \leftarrow$ *epsilon_greedy*(Q, s);
    while *episode not finished* **do**
        $(r, s') \leftarrow$ *next_state*(s, a);
        $Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s,a)\Big)$;
        $a \leftarrow$ *epsilon_greedy*(Q, s');
        $s \leftarrow s'$;

Convergence of Q-learning

Since

1. in the rhs we take $\max_{a \in \mathcal{A}} Q(s_{t+1}, a)$;
2. ε-greedy exploration guarantees to explore all the required regions of $(s, a) \in \mathcal{S} \times \mathcal{A}$;
3. $\alpha \in]0, 1[$

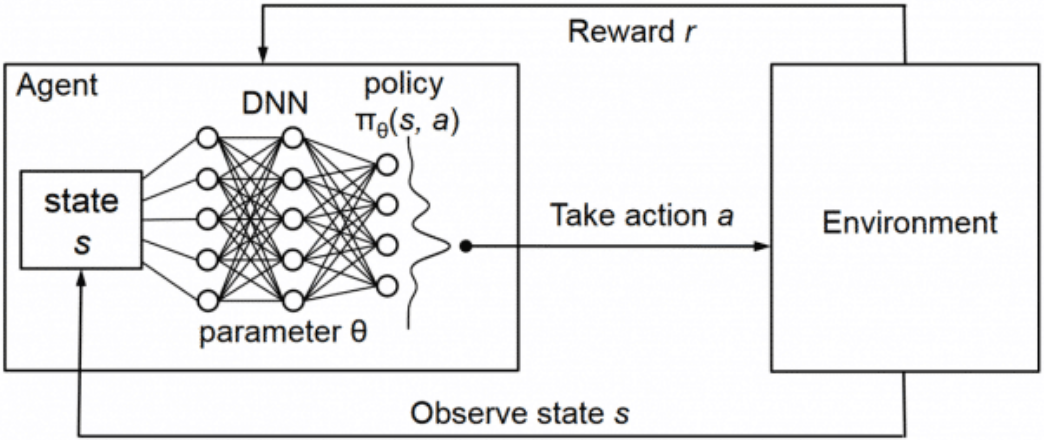
$\Longrightarrow$ the estimate of Q obtained by Q-learning converges to the optimal Q^*

Tutorial Q-learning

Let's use Q-learning!

Deep Q Network

Q-learning $\leftarrow$ unsupervised estimation of $Q(s, a)$
Starts at random $\rightarrow$ iterates $\rightarrow$ converges to the true $Q^*(s, a)$
What if $|\mathcal{S}| = +\infty$, as for example when $s \in \mathcal{S}$ can be described by a collection of *continuous* features? Q-learning stores a big table for each state... now it is not possible!
e.g. Imagine that every state a plane can assume is represented by its position in $\mathbb{R}^3$.
Idea! Instead of storing the estimates of $Q(s, a)$ in a table we can learn directly a *function approximator* that given s computes $Q(s, a)$ $\forall a$.
This can be done in continuous state spaces using a deep neural network! [MKS+15]


Reward r
Agent
DNN
policy
$\pi_\theta(s, a)$
state
s
parameter θ
Take action a
Environment
Observe state s

Training of a DQN

- Least Square regression of the TD target updating the weights θ:
  $\left[Q_{\theta_{i+1}}(s_t, a_t) - (r_t + \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a'))\right]^2 \approx 0$
  where i is an iteration index of the algorithm and t a time index along a trajectory, $i \neq t$.
- Neural Networks learn fast using batches and gradient descent. The α of Q-learning is now implemented as the learning rate of the Stochastic Gradient Descent optimizer!
- In order to use batches record (s_t, a_t, r_t, s_{t+1}) in a memory called the *Replay Buffer*.
- LS regression over a batch can be done by minimizing the MSE, however the samples must be i.i.d. This is not the case along a trajectory $\implies$ sample at random from the Replay Buffer, if it is big enough we will likely sample at each iteration i a batch containing uncorrelated transitions.

Convergence of DQN

Converges of DQN is not guaranteed since is DQN a a function approximator, but it usually happens if we use particular values for the learning rate α and the ε of an $\varepsilon-$greedy exploration strategy. It can be useful to define an exploration strategy with a decaying ε that starts very high $\varepsilon \approx 0.99$, so that the agent starts to explore the environment and little by little it will automatically focus on regions that are relevant for its learning of a good policy. More generally, [SB18] speaks of the *Deadly Triad* of Reinforcement Learning:

- Function Approximator
- Bootstrapping
- Off-policy strategy

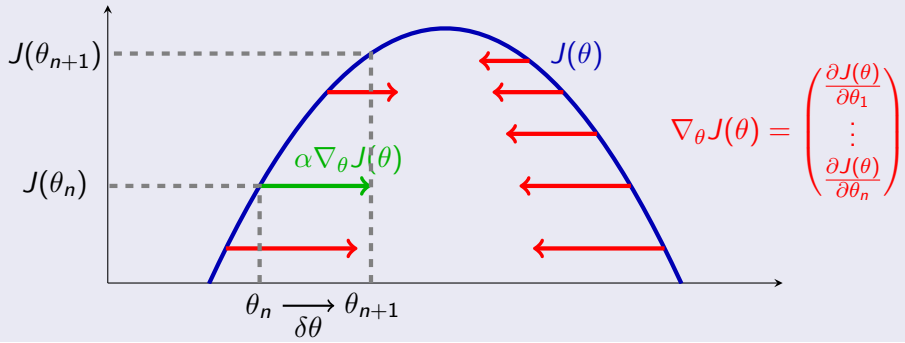
When these three factors coexist in a RL scheme its convergence can be tricky.
Power comes at a price.

And why not look directly for the strategy?

- Consider a random policy $\pi_\theta : \mathcal{S} \to [0,1]^{\mathcal{A}}$. $\forall s \in \mathcal{S}$, $\sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) = 1$.
- This policy is parameterized by $\theta \in \mathbb{R}^p$:
  the goal is then to find θ^* such that $V^{\pi_{\theta^*}}(s)$ is high $\forall s \in \mathcal{S}$.
- We then define a score $J(\theta) = \mathbb{E}[V^{\pi_\theta}(S)]$, or $J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) V^{\pi_\theta}(s)$,
  with d^{π_θ} the *stationary distribution* of the Markov Chain using π_θ
  (proportion of time spent in the state).
- We look for θ that maximizes $J(\theta)$: optimization problem!
- Use for example gradient descent.

In fact, it's a gradient ascent

The parameter update is $\delta\theta = \alpha \nabla_\theta J(\theta)$,
with $\alpha > 0$ the step-size (or learning rate) parameter, and $\nabla_\theta J(\theta)$ the gradient of the score.



J strictly concave $\Rightarrow$ returns θ^* close to the global minimum *cf.* Section 5.6.2.1 of [HMS20].
J estimated with a batch of samples $\Rightarrow$ Stochastic Gradient Descent (SGD).

Computation of the gradient

- First idea: estimation of partial derivatives (finite differences)

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \varepsilon u^{(k)}) - J(\theta)}{\varepsilon},$$

  where $u^{(k)} \in \mathbb{R}^p$, $u_i^{(k)} = \mathbb{1}_{\{i=k\}}$ and $\varepsilon > 0$.
- Given an estimation of $J(\theta)$, we just need an evaluation of the policy $\pi_{\theta+\varepsilon u^{(k)}}$.
- Algorithm: initialize θ and repeat until convergence
  - $\forall k \in \{1, \ldots, p\}$, evaluate $\pi_{\theta+\varepsilon u^{(k)}}$ by sampling trajectories
  - use evaluation $J(\theta + \varepsilon u^{(k)})$ to compute $\nabla_\theta J(\theta)$
  - update parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Computation of the gradient

- Another idea: if $\theta \mapsto \pi_\theta$ is differentiable.
- Let $\nabla_\theta \pi_\theta$ be its gradient.
- Let rewrite the score: $J(\theta) = \mathbb{E}[V^{\pi_\theta}(S)]$

$$= \mathbb{E}[Q^{\pi_\theta}(S,A)] = \mathbb{E}\Big[\mathbb{E}[Q^{\pi_\theta}(S,A) \mid S]\Big] = \mathbb{E}\left[\sum_{a\in\mathcal{A}} \pi_\theta(a \mid S)\, Q^{\pi_\theta}(S,a)\right].$$

- Its gradient becomes: $\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid S)\, Q^{\pi_\theta}(S,a)\right] =$

$$\mathbb{E}\left[\sum_{a\in\mathcal{A}} \pi_\theta(a \mid S)\, \frac{\nabla_\theta \pi_\theta(a \mid S)}{\pi_\theta(a \mid S)} Q^{\pi_\theta}(S,a)\right] = \mathbb{E}\left[\sum_{a\in\mathcal{A}} \pi_\theta(a \mid S)\, \nabla_\theta \log\Big(\pi_\theta(a \mid S)\Big) Q^{\pi_\theta}(S,a)\right]$$
$$= \mathbb{E}\Big[\mathbb{E}\left[\nabla_\theta \log\Big(\pi_\theta(A \mid S)\Big) Q^{\pi_\theta}(S,A) \,\middle|\, S\right]\Big] = \mathbb{E}\left[\nabla_\theta \log\Big(\pi_\theta(A \mid S)\Big) Q^{\pi_\theta}(S,A)\right].$$

Monte-Carlo Policy Gradient (REINFORCE)

By using the return $G(S_t, A_t) = R_t + \gamma R_{t+1} + \ldots + \gamma^{T-t} R_T$ as estimate for $Q(S_t, A_t)$,

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \Big(\pi_\theta \left(a \mid s \right) \Big) G(s, a \Big).$$

---

input : step size α, number of episodes n
output: optimized θ
initialise θ arbitrarily;
for i **in** $\{1, \ldots, n\}$ **do**
  generate $(s_0, s_1, \ldots, s_T)$ using π_θ;
  for t **in** $\{0, \ldots, T\}$ **do**
    $N(s_t) \leftarrow N(s_t) + 1$;
    $G(s_t, a_t) \leftarrow \sum_{i=t}^{T} \gamma^{i-t} r\Big(s_i, \pi(s_i)\Big)$;
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \Big(\pi_\theta \left(a_t \mid s_t \right) \Big) G(s_t, a_t)$;

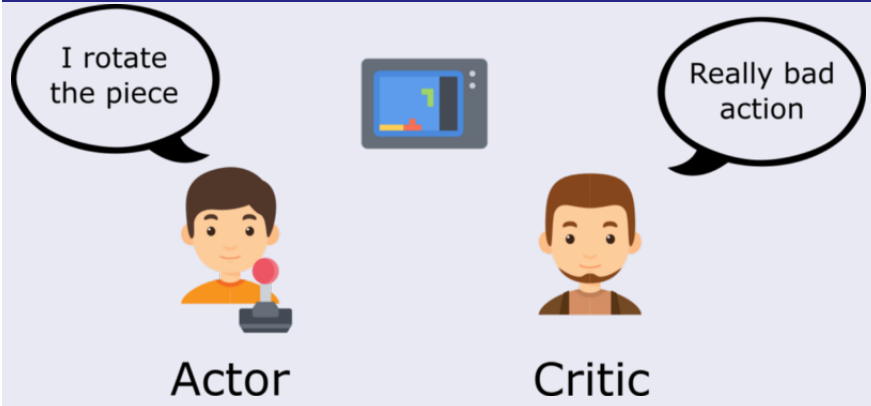
Reducing variance using a critic

- Monte-Carlo policy gradient still has high variance
- We use a **critic** to estimate the action-value function $Q_\eta \approx Q^{\pi_\theta}$, with $\eta \in \mathbb{R}^q$ containing other parameters
- Actor-critic algorithms maintain two sets of parameters
  - Critic updates action-value function parameters η
  - Actor updates policy parameters θ, in direction suggested by critic
- Actor-critic algorithms follow an approximate policy gradient

$$\begin{aligned} \nabla_\theta J(\theta) &\approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log\Big(\pi_\theta(a \mid s)\, Q_\eta(s,a)\Big)\right]. \\ \delta\theta &= \alpha \nabla_\theta \log\Big(\pi_\theta(a \mid s)\Big) Q_\eta(s,a). \end{aligned}$$

See also this introduction to Advantage Actor Critic.

Actor Critic



Institut Supérieur de l'Aéronautique et de l'Espace

10 avenue Édouard Belin – BP 54032

31055 Toulouse Cedex 4 – France

Phone: +33 5 61 33 80 80

www.isae-supaero.fr

isae
Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO