# Reinforcement Learning

## Lecture 1 : Markov Decision Processes

Emilie Kaufmann
(emilie.kaufmann@univ-lille.fr)




Ecole Centrale de Lille, 2021/2022

# Practical information

- webpage of the class :
  https://emiliekaufmann.github.io/teaching.html
  (slides online before the class)

- 3 pratical sessions with Omar Darwiche Domingues :
  - ➜ Python and jupyter notebook
  - ➜ open AI gym (to create RL environements)
  - ➜ RL berry (to try and evaluate RL algorithms)
  - ➜ Pytorch (for Deep RL)

- Evaluation of the class :
  - ➜ the last practical session will be graded
  - ➜ project (paper reading) : report + (visio) presentation

# What is Reinforcement Learning ?

- learning by "trial and error"
- learning to behave in an unknown, shochastic environement by maximizing some real-valued reward signal

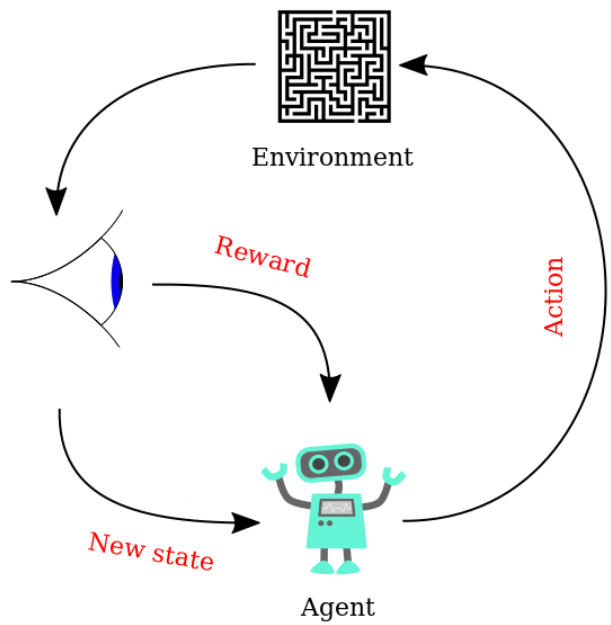

*Example : learning to bike without a perfect knowledge of physics*

# Key RL concepts

A learning agent *sequentially* interacts with its environment by performing actions. Each action

- provides an instantaneous reward
- leads to an evolution of the agent's state

**Agent's goal :** act so as to maximize its total reward



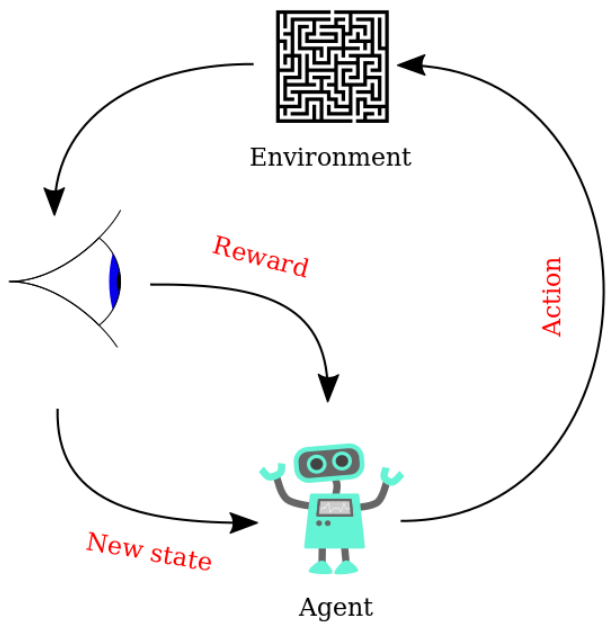

source : Wikipedia

# Key RL concepts

Keywords (high-level) :

- Reward : instantaneous feedback received after acting
- Policy : strategy to choose an action in a given state
- Value : total reward the agent can get in some state by following some policy

**Agent's goal :** find a policy that maximizes the value in each state




source : Wikipedia

# RL successes : Games (1/2)

From Backgammon...

1992, TD-gammon

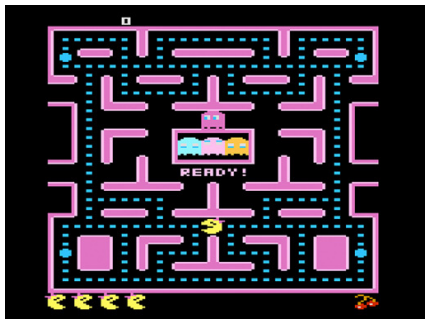
... to Go

2015, AlphaGo
2017, AlphaGo Zero

➜ RL agents learn new types of strategies

# RL successes : Games (2/2)

- Learning to play from pixels (and rewards) : Atari Games 2010+ Deep Reinforcement Learning





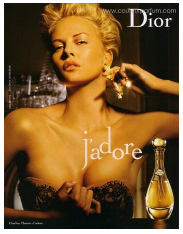
- Recent challenges : multi-player / partial information games

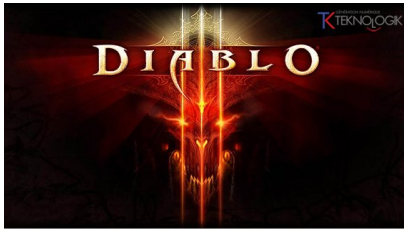

OpenAI Five (2019)

Pluribus (2019)

# RL sucessess : Content Optimization

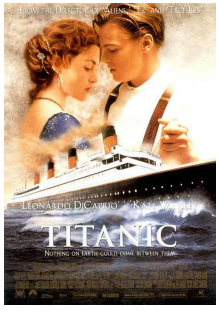
- online advertisement

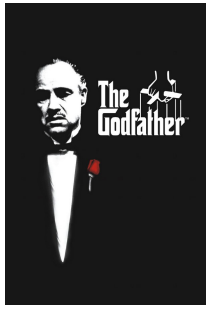


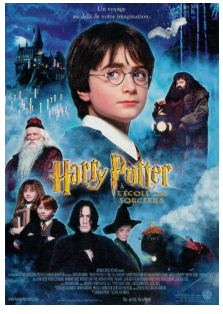


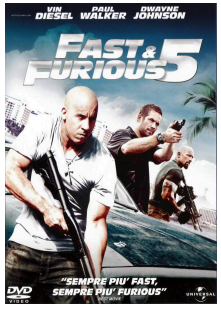

➜ action : display an add / reward : click

- (sequential) recommender systems





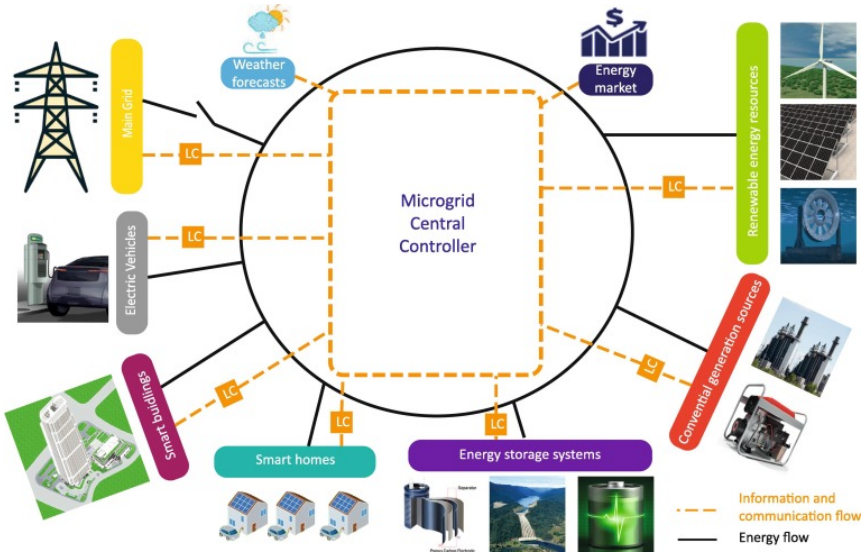





➜ action : recommend a movie / reward : rating

# RL : Many potential applications

- Smart grid / microgrid management




source : ScienceDirect.com

Actions :

- charge or discharge storage systems
- turn on or off renewable energy source
- buy energy from the market

...

Reward : - Cost

# RL : Many potential applications

- Autonomous robotics

- Self-driving cars ?

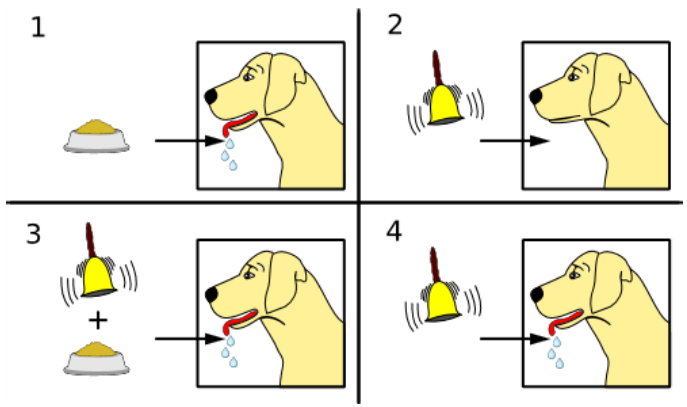
# History of RL

- Learning to behave from rewards : an old idea from psychology

- 1900s : observation of animal behavior
  (e.g. Thorndike 1911 "Law of Effect")

*Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will [...] be more likely to recur.*

- 1920s : Pavlov work on conditionnal reflexes
  first occurence of "reinforcement" in animal learning




source : Wikipedia

# History of RL

- Learning to behave from rewards : an inspiration from the brain ?

- 1950s : first experiments on electric brain stimuli for controlling mice behavior (Oak and Miller 1954)

➜ hypothesis that dopamine broadcast rewards signal to the brain

# History of RL

- Some steps towards computational RL

- 1950s, Shannon's machines : "Theseus", a mice finding how to get out of a maze, a chess player, a Rubik's cube solver
- 1957, Bellmann : Dynamic Programming (control of dynamical systems)
- 1961, Minsky "Towards artificial intelligence"
- 1978, Sutton : Temporal Difference Learning (artificial intelligence)
- 1989, Watkins : Q-Learning algorithm

Nowadays, reinforcement learning is mostly formalized as learning an optimal policy in an incompletely-known Markov Decision Process.

# Markov Decision Process

A Markov Decision Process (MDP) models a situation in which repeated decisions (= choices of actions) are made. MDP provides models for the consequence of each decisions :

- in terms of reward
- in terms of the evoluation of the system's state

In each (discrete) decision time $t = 1, 2, \dots,$ a learning agent

- selects an action $a_t$ based on his current state $s_t$ (or possibly all the previous observations),
- gets a reward $r_t \in \mathbb{R}$ depending on his choice,
- transits to a new state $s_{t+1}$ depending on his choice.

# Markov Decision Process

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

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu_{(s,a)})_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the reward distribution for the state-action pair $(s, a)$
- $P = (p(\cdot|s, a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair $(s, a)$

In each (discrete) decision time $t = 1, 2, \ldots,$ a learning agent

- selects an action $a_t$ based on his current state $s_t$ (or possibly all the previous observations),
- gets a reward $r_t \sim \nu_{(s_t,a_t)}$
- makes a transition to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$

[Bellman 1957, Howard 1960, Blackwell 70s...]

# Markov Decision Process

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

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

In each (discrete) decision time $t = 1, 2, \ldots,$ a learning agent

- selects an action $a_t$ based on his current state $s_t$ (or possibly all the previous observations),
- gets a reward $r_t \sim \nu_{(s_t,a_t)}$
- makes a transition to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$

[Bellman 1957, Howard 1960, Blackwell 70s...]

# Markov Decision Process

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

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu_{(s,a)})_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the reward distribution for the state-action pair $(s, a)$
- $P = (p(\cdot|s, a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair $(s, a)$

**Goal** : (made more precise later) select actions so as to maximize some notion of *expected* cumulated rewards

## Mean reward of action $a$ in state $s$

$$r(s, a) = \mathbb{E}_{R\sim\nu_{(s,a)}}[R]$$

# Markov Decision Process

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

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu_{(s,a)})_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the reward distribution for the state-action pair $(s, a)$
- $P = (p(\cdot|s, a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair $(s, a)$

- **The tabular case** : finite state and action spaces

$$\begin{aligned} \mathcal{S} &= \{1, \dots, S\} \\ \mathcal{A} &= \{1, \dots, A\} \end{aligned}$$

For every $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $p(s'|s, a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$.

# Why Markov ?

In an MDP, the sequence of sucessive states / actions / rewards

$$s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t$$

satisfies some extension of the Markov property :

$$\begin{aligned}\mathbb{P}\left(s_t = s, r_{t-1} = r | s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}\right) \\ = \mathbb{P}\left(s_t = s, r_{t-1} = r | s_{t-1}, a_{t-1}\right)\end{aligned}$$

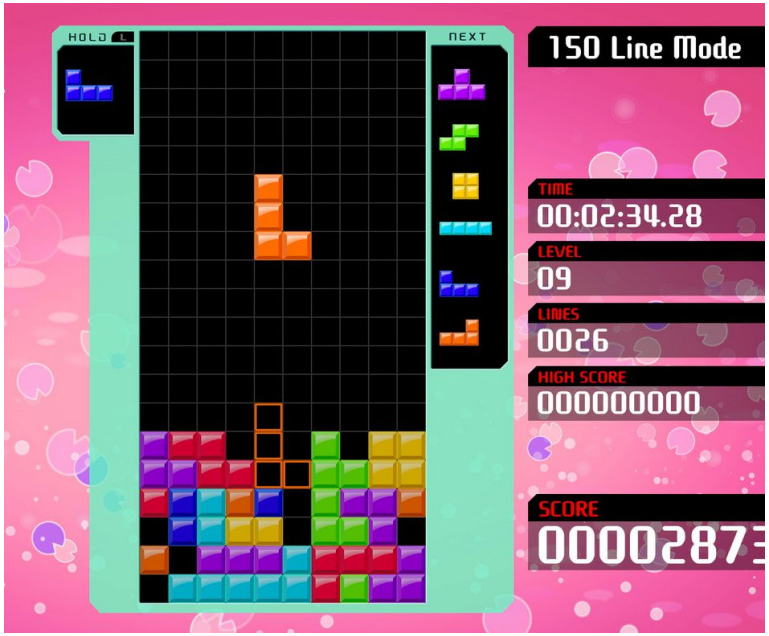
*(discrete action and reward)*

## Definition

A Markov chain on a discrete space $\mathcal{X}$ is a stochastic process $(X_t)_{t\in\mathbb{N}}$ that satisfies the Markov property :

$$\mathbb{P}(X_t = x_t | X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = \mathbb{P}(X_t = x_t | X_{t-1} = x_{t-1}).$$

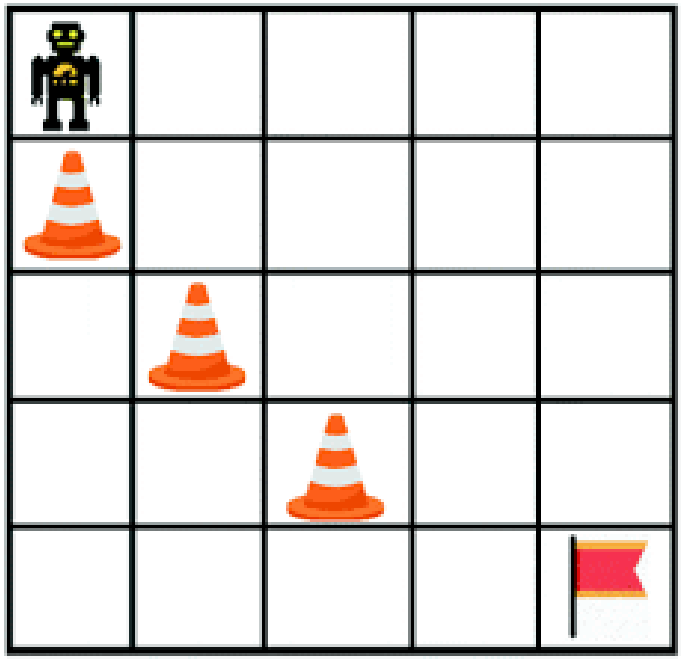
# Example : Tetris




- **State** : current board and next blocks to add
- **Action** : orientation + position of the dropped block
- **Reward** : increment in the score/ number of lines
- **Transition** : new board + randomness in the new block

➜ difficulty : large state space !

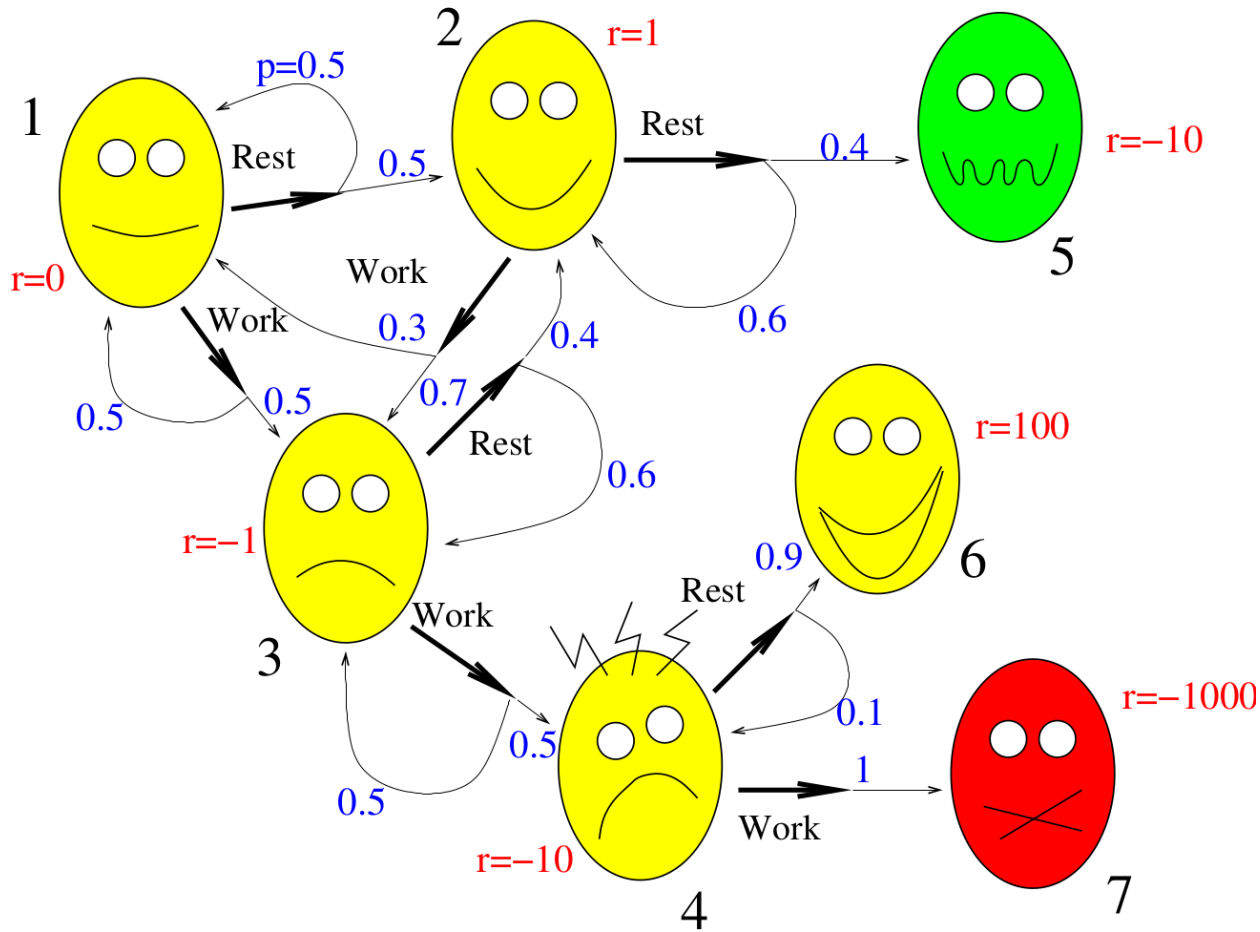
# Example : Grid world



- **State** : position of the robot
- **Actions** : $\leftarrow, \uparrow, \rightarrow, \downarrow$
- **Transitions** : (quasi) deterministic
- **Rewards** : depends on the behavior to incentivise (positive or negative rewards on some states / $-1$ for each step before a goal...)

➜ possible difficulty : sparse rewards

# Example : The Student Dilemma




credit : Rémi Munos, Alessandro Lazaric

# (running) Example : Retail Store Management

You own a bike store. During week $t$, the (random) demand is $D_t$ units.
On Monday morning you may choose to command $a_t$ additional units :
they are delivered immediately before the shop opens.

For each week :

- Maintenance cost : $h$ per unit left in your stock
- Ordering cost : $c$ per unit ordered + fix cost $c_0$ if an order is placed
- Sales profit : $p$ per unit sold

Constraints :

- your warehouse has a maximal capacity of $M$ bikes
  (any additional bike gets stolen)
- you cannot sell bikes that you don't have in stock

Exercise : Write down the underlying Markov Decision Process

# Policies

## Definition

A (Markovian) policy is a sequence $\pi = (\pi_t)_{t \in \mathbb{N}^*}$ of mappings

$$\pi_t : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\Delta(\mathcal{A})$ is the set of probability distributions over the action space.

➜ An agent acting under policy $\pi$ selects at round $t$ the action

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

# Policies

## Definition

A (Markovian) policy is a sequence $\pi = (\pi_t)_{t \in \mathbb{N}^*}$ of mappings

$$\pi_t : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\Delta(\mathcal{A})$ is the set of probability distributions over the action space.

- An agent acting under policy $\pi$ selects at round $t$ the action

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

- **Remark** : one could also consider *history-dependent* policies $\pi_t : \mathcal{H}_t \to \Delta(\mathcal{A})$, where the next action is chosen based on

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

# Policies

## Definition

A (Markovian) policy is a sequence $\pi = (\pi_t)_{t \in \mathbb{N}^*}$ of mappings

$$\pi_t : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\Delta(\mathcal{A})$ is the set of probability distributions over the action space.

A policy may be

| **Deterministic** | **Stochastic** |
|---|---|
| $\pi_t : \mathcal{S} \to \mathcal{A}$ | $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$ |

- **Terminology** : policy = strategy = decision rule = control

# Policies

## Definition

A (Markovian) policy is a sequence $\pi = (\pi_t)_{t \in \mathbb{N}^*}$ of mappings

$$\pi_t : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\Delta(\mathcal{A})$ is the set of probability distributions over the action space.

A policy may be

| **Stationary** | **Non-stationary** |
|---|---|
| $\pi = (\pi, \pi, \pi, \dots)$ | $\pi = (\pi_1, \pi_2, \dots)$ |

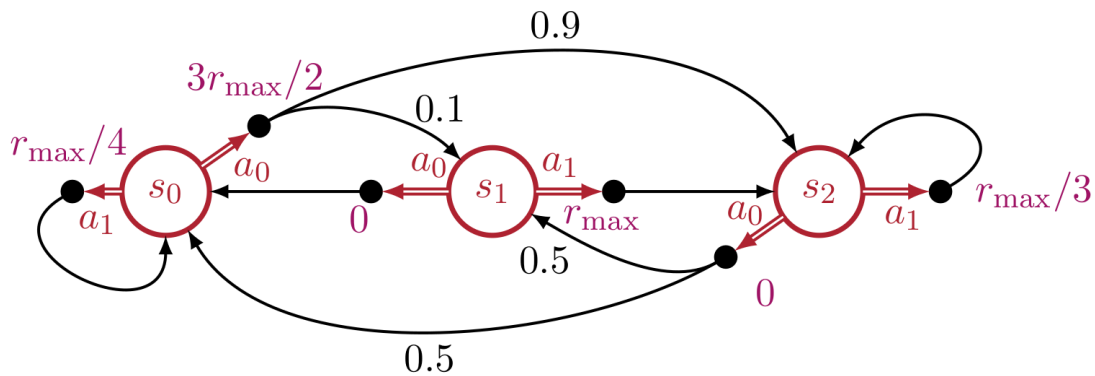
- **Terminology** : policy = strategy = decision rule = control

# Policies

Under a stationary (deterministic) policy $\pi : \mathcal{S} \to \mathcal{A}$, the random process $(s_t)_{t\in\mathbb{N}}$ is a Markov chain, with transition probability

$$\mathbb{P}^\pi(s_{t+1} = s' | s_t = s) = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = \pi(s)) = p(s' | s, \pi(s))$$

*(can be extended to stochastic policies and continuous spaces)*



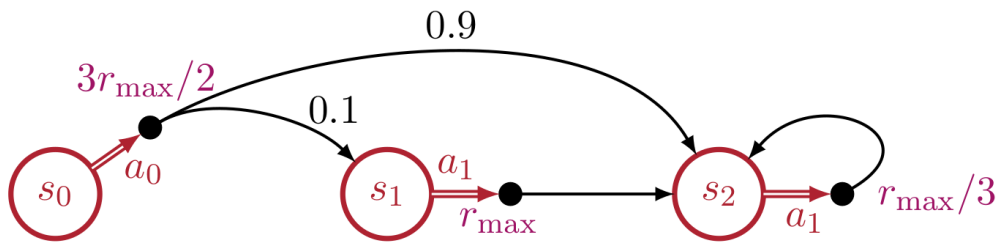

➜ A MDP is sometimes referred to as a controlled Markov chain

# Policies

Under a stationary (deterministic) policy $\pi : \mathcal{S} \to \mathcal{A}$, the random process $(s_t)_{t\in\mathbb{N}}$ is a Markov chain, with transition probability

$$\mathbb{P}^\pi(s_{t+1} = s' | s_t = s) = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = \pi(s)) = p(s' | s, \pi(s))$$

*(can be extended to stochastic policies and continuous spaces)*




➜ A MDP is sometimes referred to as a controlled Markov chain

# Value function

## Value of a policy $\pi$ in a state $s \in \mathcal{S}$

$V^\pi(s)$ measures the **expected** cumulative reward obtained by an agent starting from state $s$ and applying policy $\pi$.

➜ $\neq$ notions of cumulative reward provide $\neq$ definitions of the value

① **Finite horizon**

Given a known horizon $H \in \mathbb{N}^*$,

$$V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=1}^{H} r_t \middle| s_1 = s \right]$$

# Value function

## Value of a policy $\pi$ in a state $s \in \mathcal{S}$

$V^\pi(s)$ measures the **expected** cumulative reward obtained by an agent starting from state $s$ and applying policy $\pi$.

➜ $\neq$ notions of cumulative reward provide $\neq$ definitions of the value

① **Finite horizon**

Given a known horizon $H \in \mathbb{N}^*$,

$$V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=1}^{H} r_t \middle| s_1 = s \right]$$

$a_t \sim \pi(s_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, $r_t \sim \nu_{s_t,a_t}$

# Value function

## Value of a policy $\pi$ in a state $s \in \mathcal{S}$

$V^\pi(s)$ measures the **expected** cumulative reward obtained by an agent starting from state $s$ and applying policy $\pi$.

➜ $\neq$ notions of cumulative reward provide $\neq$ definitions of the value

① **Finite horizon**

Given a known horizon $H \in \mathbb{N}^*$,

$$V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=1}^{H} r_t \,\middle|\, s_1 = s \right]$$

$a_t \sim \pi(s_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, $r_t \sim \nu_{s_t,a_t}$

starting from state $s$

➜ **When is it used ?** In the presence of a natural notion of duration of an episode (e.g. maximal number of steps in a game)

# Value function

## Value of a policy $\pi$ in a state $s \in \mathcal{S}$

$V^\pi(s)$ measures the **expected** cumulative reward obtained by an agent starting from state $s$ and applying policy $\pi$.

➜ $\neq$ notions of cumulative reward provide $\neq$ definitions of the value

② **Infinite time horizon with a discount parameter**

Given a known discount parameter $\gamma \in (0,1)$,

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

$a_t \sim \pi(s_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, $r_t \sim \nu_{s_t, a_t}$

starting from state $s$

# Value function

## Value of a policy $\pi$ in a state $s \in \mathcal{S}$

$V^\pi(s)$ measures the **expected** cumulative reward obtained by an agent starting from state $s$ and applying policy $\pi$.

➜ $\neq$ notions of cumulative reward provide $\neq$ definitions of the value

② **Infinite time horizon with a discount parameter**

Given a known discount parameter $\gamma \in (0,1)$,

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

$a_t \sim \pi(s_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, $r_t \sim \nu_{s_t, a_t}$

starting from state $s$

➜ **When is it used ?** To put more weight on short-term reward / when there is a natural notion of discount

# Other possible definitions

**(not discussed much in this class)**

③ **Infinite time horizon with a terminal state**

Given $\tau$ the random time at which we first reach a terminal state.

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=1}^{\tau} r_t \middle| s_1 = s\right]$$

➜ **When ?** For tasks that have a natural notion of terminal state

④ **Infinite time horizon with average reward**

$$V^{\pi}(s) = \lim_{T\to\infty} \mathbb{E}^{\pi}\left[\frac{1}{T}\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

➜ **When ?** The system should be controlled for a very long time

# Optimal policy

Given a **value function** (①,②,③ or ④), one can define the following.

## Definition

The optimal value in a state $s$ is given by

$$V^\star(s) = \max_\pi V^\pi(s).$$

## Theorem [Puterman, 1994]

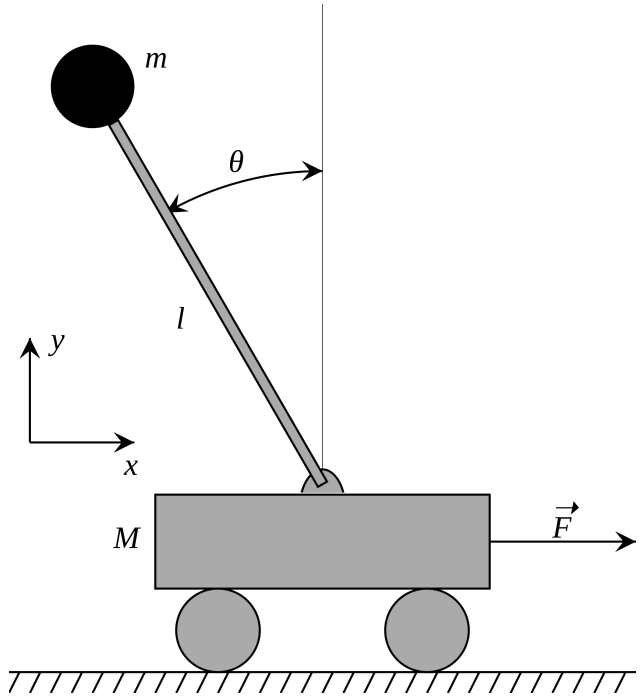
There exists an optimal policy $\pi^\star$ which satisfies

$$\forall s \in \mathcal{S},\ \pi^\star \in \operatorname*{argmax}_\pi V^\pi(s)$$

Therefore, one can write $V^\star = V^{\pi^\star}$.

➜ as we shall see, one of these optimal policies is deterministic .

# Let's try a policy on Cart Pole




**Task :** maintain the pole as long as possible in a quasi-vertical position, by applying some force on the cart towards the left or right

one of the classical environement in OpenAI Gym

see the introductory notebook

# Back to Retail Store Management

- State : number of bikes in stock on Sunday
  State space : $\mathcal{S} = \{0, \dots, M\}$
- Action : number of bikes ordered at the beginning of the week
  Action space : $\mathcal{A} = \{0, \dots, M\}$
- Reward = balance of the week : if your stock was $s_t$ and you order $a_t$ bikes, in week $t$ you earn

$$r_t = -c_0 \mathbb{1}_{(a_t>0)} - c \times a_t - h \times s_t + p \times \min(D_t, s_t + a_t, M)$$

- Transition : you end the week with

$$s_{t+1} = \max\left(0, \min(M, s_t + a_t) - D_t\right) \qquad \text{bikes}$$

**Goal :** From an initial stock $s$, maximize the sum of discounted rewards

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

# Possible policies

- Uniform policy :
$$\pi(s) \sim \mathcal{U}(\{0, \dots, M - s\})$$
- Constant policy : always buy $m_0$ bikes
$$\pi(s) = \max(M - s, m_0)$$
- Threshold policy : whenever there are less than $m_1$ bikes in stock, refill it up to $m_2$ bikes. Otherwise, do not order.
$$\pi(s) = \mathbb{1}_{(s \leq m_1)}(m_2 - s)$$

# Simulations

Let's try out some policies !

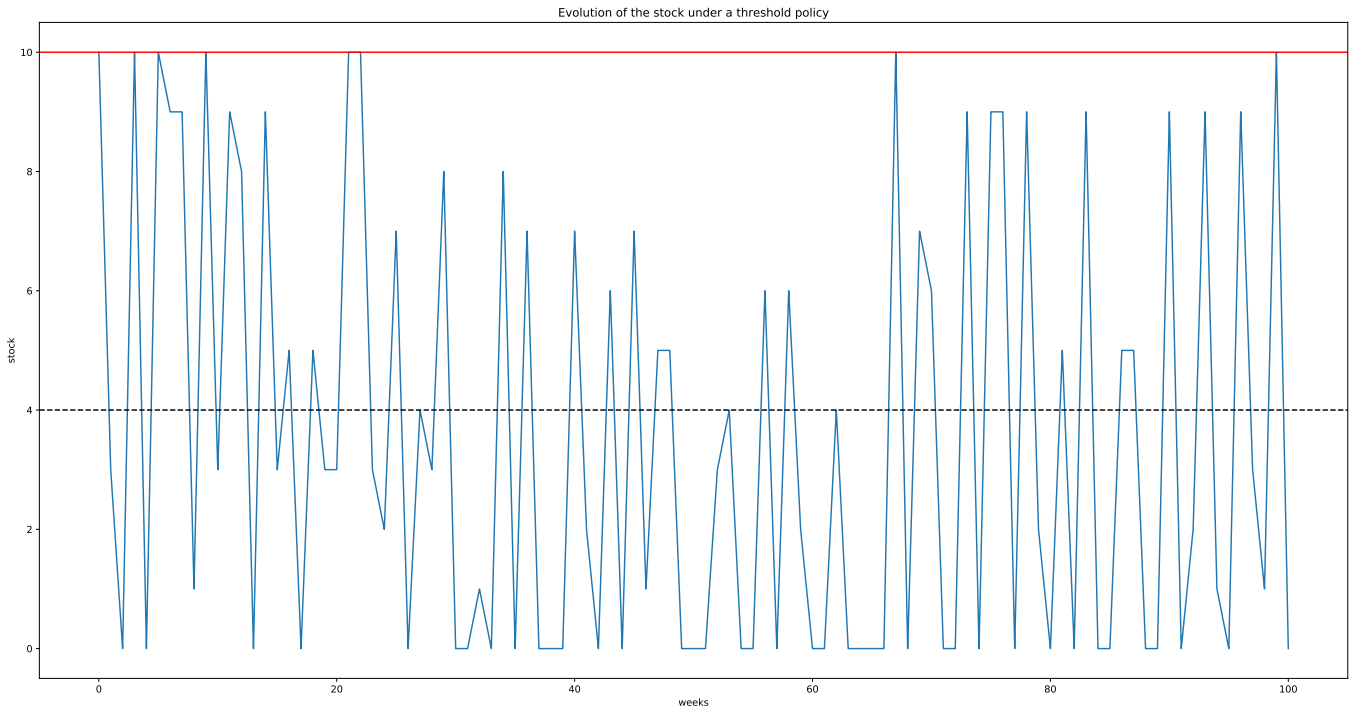



FIGURE – Evolution of the stock $s_t$ under a threshold policy ($m_1 = 4, m_2 = 10$)

# Questions

In an known Markov Decision Process

- can we compute an optimal policy ?
  (based on the explicit knowledge of $r(s,a)$ and $p(\cdot|s,a)$)
- ... even with very large (or infinite) state and/or action spaces ?
  (e.g. based on a *simulator* for transitions)

Beyond :

- Can we learn a good policy in an unknown MDP, only by selecting actions and performing transitions ?
- ... and can we do it while maximizing reward ?

# Questions

In an known Markov Decision Process

- can we compute an optimal policy ?
  (based on the explicit knowledge of $r(s,a)$ and $p(\cdot|s,a)$)
- ... even with very large (or infinite) state and/or action spaces ?
  (e.g. based on a *simulator* for transitions)

Beyond :

- Can we learn a good policy in an unknown MDP, only by selecting actions and performing transitions ?
- ... and can we do it while maximizing reward ?

**Broad goal of Reinforcement Learning**

Learning an optimal **policy** in an unknown (or very large) MDP, by acting (=choosing action) and observing transitions.

# Policy evaluation

Given a policy $\pi = (\pi_t)_{t\in\mathbb{N}}$, how can we compute

- Finite horizon MDP :

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

  and in particular $V^\pi(s) = V_1^\pi(s)$

- Discounted MDP :

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

# Intermezzo : Probability theory

We will need to compute several conditional expectations.

Recall that :

- $\mathbb{E}[X|Y=y]$ is a number :

$$\mathbb{E}[X|Y=y] = \sum_{x \in \mathcal{X}} x \mathbb{P}(X=x|Y=y) \text{ in the discrete case}$$

- $\mathbb{E}[X|Y]$ is a random variable that is $\sigma(Y)$-measurable

$$\mathbb{E}[X|Y] = \sum_{y \in \mathcal{Y}} \mathbb{1}_{(Y=y_i)} \mathbb{E}[X|Y=y_i] \text{ in the discrete case}$$

- more generally $\mathbb{E}[X|\mathcal{F}]$ is random variable that is $\mathcal{F}$-measurable

## Useful properties

- Law of total expectation : $\mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}[X]$.
- $\mathbb{E}[X] = \sum_{y \in \mathcal{Y}} P(Y=y)\mathbb{E}[X|Y=y]$.

# Intermezzo : Probability theory

We will need to compute several conditional expectations.

Recall that :

- $\mathbb{E}[X|Y=y]$ is a number :

$$\mathbb{E}[X|Y=y] = \sum_{x \in \mathcal{X}} x \mathbb{P}(X=x|Y=y) \text{ in the discrete case}$$

- $\mathbb{E}[X|Y]$ is a random variable that is $\sigma(Y)$-measurable

$$\mathbb{E}[X|Y] = \sum_{y \in \mathcal{Y}} \mathbb{1}_{(Y=y_i)} \mathbb{E}[X|Y=y_i] \text{ in the discrete case}$$

- more generally $\mathbb{E}[X|\mathcal{F}]$ is random variable that is $\mathcal{F}$-measurable

## Useful properties

- Law of total expectation : $\mathbb{E}[\mathbb{E}[X|\mathcal{F}]] = \mathbb{E}[X]$.
- $\mathbb{E}[X] = \sum_{y \in \mathcal{Y}} P(Y=y)\mathbb{E}[X|Y=y]$.

# Bellman equations (finite horizon)

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

## Proposition

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

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

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

Exercise : Prove it !


Puterman, M. (1994).
*Markov Decision Processes. Discrete Stochastic. Dynamic Programming.*
Wiley.