Centre de Recherche en Informatique,
Signal et Automatique de Lille

Reinforcement Learning
From Control to Learning

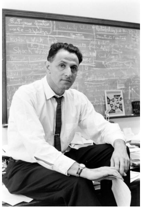
Odalric-Ambrym Maillard

2021-2022

Inria Scool

Table of contents

MDPs were popularized in the 1950s by Richard Bellman 1957, Ronald Howard 1960 (following works from Massé, Arrow, etc.)




Appear in various fields of research:

- **Control theory**: Stochastic optimal control.
- **Operations research**: Stochastic shortest paths.
- **Machine Learning**: Reinforcement learning (RL).
- **Economics**: Sequential decision under uncertainty.

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.
The system is described by state variables, forming the **state space** $\mathcal{S}$.

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

The system is described by state variables, forming the **state space** $\mathcal{S}$.

- $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.

The decision-maker chooses **actions** at some **decision steps**.

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

The system is described by state variables, forming the **state space** $\mathcal{S}$.

- $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.

The decision-maker chooses **actions** at some **decision steps**.

- A set of actions $\mathcal{A}_s$ is available in each state $s \in \mathcal{S}$.

(Generic) Markov Decision Processes

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

The system is described by state variables, forming the **state space** $\mathcal{S}$.

- ▷ $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.

The decision-maker chooses **actions** at some **decision steps**.

- ▷ A set of actions $\mathcal{A}_s$ is available in each state $s \in \mathcal{S}$.
- ▷ The set of decision steps $\mathbb{T}$ can be discrete or continuous.

Choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$ affects the evolution of the system, hence the next states.

(Generic) Markov Decision Processes

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

The system is described by state variables, forming the **state space** $\mathcal{S}$.

- $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.

The decision-maker chooses **actions** at some **decision steps**.

- A set of actions $\mathcal{A}_s$ is available in each state $s \in \mathcal{S}$.
- The set of decision steps $\mathbb{T}$ can be discrete or continuous.

Choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$ affects the evolution of the system, hence the next states.

- The system may **transit** to a new state only at decision steps (rested) or at any time (restless).

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.

The system is described by state variables, forming the **state space** $\mathcal{S}$.

- ▷ $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.

The decision-maker chooses **actions** at some **decision steps**.

- ▷ A set of actions $\mathcal{A}_s$ is available in each state $s \in \mathcal{S}$.
- ▷ The set of decision steps $\mathbb{T}$ can be discrete or continuous.

Choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$ affects the evolution of the system, hence the next states.

- ▷ The system may **transit** to a new state only at decision steps (rested) or at any time (restless).
- ▷ The new state is generated according to some distribution $\mathbf{p}_t(s, a) \in \mathcal{P}(\mathcal{S})$.

To guide the decision-maker, a real-value called the **reward** is generated when choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$:

(Generic) Markov Decision Processes

A **Markov Decision Process** (MDP) is a formalization of a **decision-maker** interacting with a **dynamical system**.
The system is described by state variables, forming the **state space** $\mathcal{S}$.

- $\mathcal{S}$ can be discrete (finite or countable) or continuous (bounded or not).

The system evolves with time. At time $t \in \mathbb{R}$, the system is in state $s_t \in \mathcal{S}$.
The decision-maker chooses **actions** at some **decision steps**.

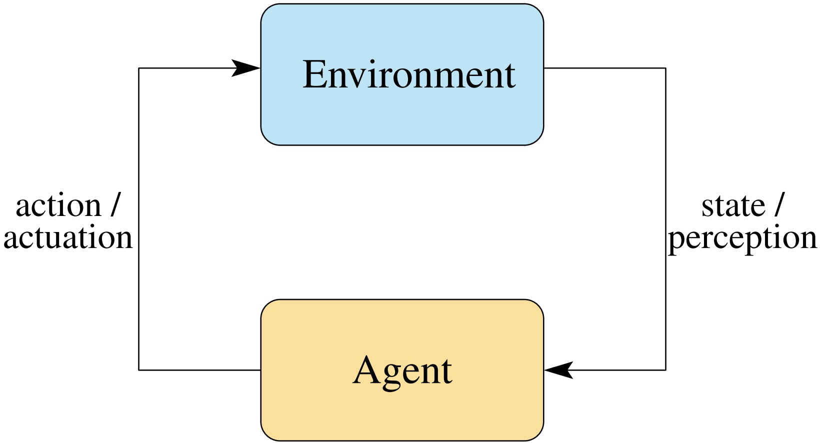
- A set of actions $\mathcal{A}_s$ is available in each state $s \in \mathcal{S}$.
- The set of decision steps $\mathbb{T}$ can be discrete or continuous.

Choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$ affects the evolution of the system, hence the next states.

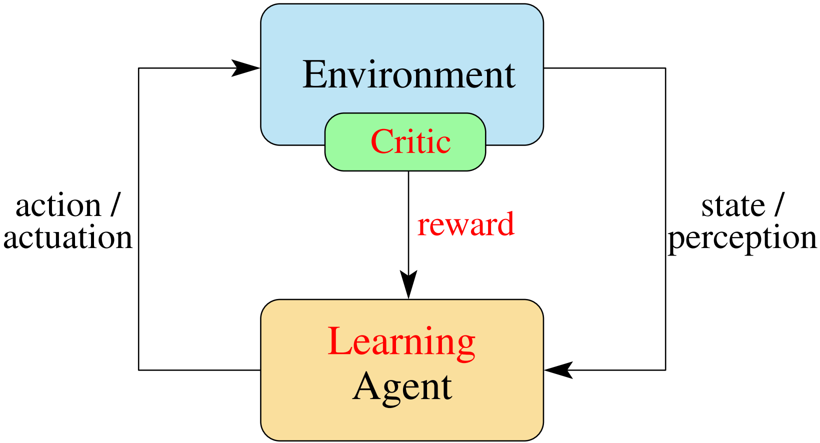
- The system may **transit** to a new state only at decision steps (rested) or at any time (restless).
- The new state is generated according to some distribution $\mathbf{p}_t(s, a) \in \mathcal{P}(\mathcal{S})$.

To guide the decision-maker, a real-value called the **reward** is generated when choosing action $a \in \mathcal{A}_s$ in state s at decision-step $t \in \mathbb{T}$:

- The reward is generated according to some distribution $\mathbf{r}_t(s, a) \in \mathcal{P}(\mathbb{R})$.


Environment
action /
actuation
state /
perception
Agent

The Reinforcement Learning Model



for $t = 1, \dots, n$ **do**

    The agent perceives state s_t

    The agent performs action a_t

    The environment evolves to s_{t+1}

    The agent receives reward r_t

end for

MDPs

In the sequel, we focus on **discrete-time, rested, stationary, fully observed** MDPs: $\mathbb{T} = \mathbb{N}$, $\mathbf{p}_t$, $\mathbf{r}_t$ are independent on time, S_t, R_t are known to the agent.

- A (discrete-time, rested, stationary) **Markov Decision Process** (MDP) is a tuple $M = (\mathcal{S}, \mathcal{A}, \mathbf{r}, \mathbf{i}, \mathbf{p})$ with
  - **State** space $\mathcal{S}$, **Action** space $\mathcal{A}$ (Actions $\mathcal{A}_s = \mathcal{A}$ available in state $s \in \mathcal{S}$),
  - **Transition** distribution: $\forall s \in \mathcal{S}, a \in \mathcal{A}_s, \quad \mathbf{p}(s,a) \in \mathcal{P}(\mathcal{S})$,
  - **Reward** distribution: $\forall s \in \mathcal{S}, a \in \mathcal{A}_s, \quad \mathbf{r}(s,a) \in \mathcal{P}(\mathbb{R})$ with mean $\mathbf{m}(s,a)$.
- An agents acts (chooses an action in some state) at time t according to a **policy**:

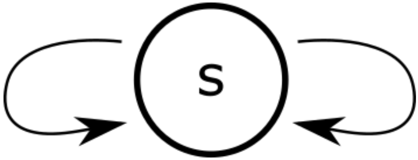
	stationary	history-dependent
deterministic	$\pi : \mathcal{S} \to \mathcal{A}$	$\pi_t : \mathcal{H}_t \to \mathcal{A}$
stochastic	$\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$	$\pi_t : \mathcal{H}_t \to \mathcal{P}(\mathcal{A})$.

- Starting from initial state $s_1 \sim \mathbf{i} \in \mathcal{P}(\mathcal{S})$, the process generates history

$$H_t = (S_1, A_1, A_1, \dots, A_{t-1}, R_{t-1}, S_t)$$

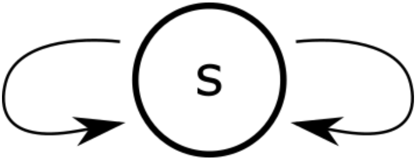
with $A_t \sim \pi_t(H_{t-1})$, $R_t \sim \mathbf{r}(S_t, A_t)$, $S_{t+1} \sim \mathbf{p}(S_t, A_t)$.

Simplest MDP model



Actions: $\{\leftarrow, \rightarrow\}$
Rewards: $\mathbf{m}(s, \leftarrow) = m_1, \mathbf{m}(s, \rightarrow) = m_2$.

Simplest MDP model



Actions: $\{\leftarrow, \rightarrow\}$
Rewards: $\mathbf{m}(s, \leftarrow) = m_1, \mathbf{m}(s, \rightarrow) = m_2$.

Maximizing rewards on average is trivial when $\mathbf{r}$ is **known**: action with highest mean.
Much more challenging when $\mathbf{r}$ is **unknown**: See **Bandit** Lectures.

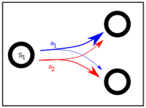
▷ In **Control Theory**, $(\mathcal{S}, \mathcal{A}, \mathbf{r}, \mathbf{p})$ are assumed perfectly known by the decision-maker.

▷ In **Reinforcement Learning**, parts of $(\mathcal{S}, \mathcal{A}, \mathbf{r}, \mathbf{p})$ may not be known ahead of time, hence must be learned from interactions:

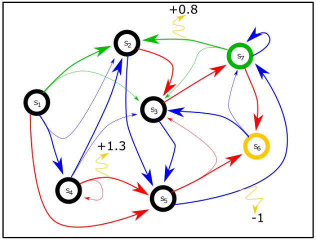
Formalism	Known/Observed	Unknown
Control	$\mathcal{S}, \mathcal{A}, \mathbf{r}, \mathbf{p}, S_t, R_t$	$\emptyset$
Fully-observed (MDP)	$\mathcal{S}, \mathcal{A}, S_t, R_t$	$\mathbf{r}, \mathbf{p}$
Partially-observed (PO-MDP)	$\mathcal{S}, \mathcal{A}, R_t$	$\mathbf{r}, \mathbf{p}, S_t$
Predictive-state representations (PSR)	$\mathcal{A}, R_t$	$\mathbf{r}, \mathbf{p}, \mathcal{S}, S_t$

▷ In **Inverse Reinforcement Learning**, $\mathbf{r}$ is unknown. We observe **trajectories** of interaction between a policy and a system.

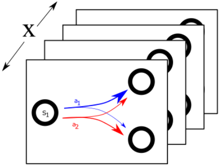
Bandit



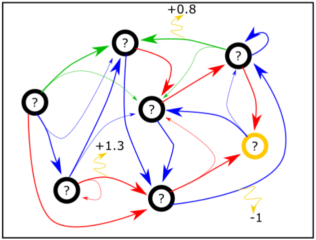
MDP



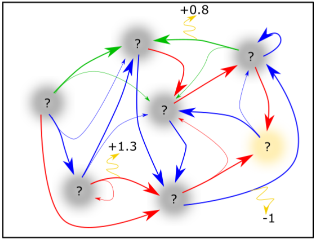
Contextual bandit



PO-MDP



PSR



▷ The discrete-time dynamic system generating random variable $(X_t)_{t\in\mathbb{N}}$ is a **Markov chain** if it satisfies the **Markov property**

$$\mathbb{P}(X_{t+1} = x \,|\, X_t, X_{t-1}, \ldots, X_0) = \mathbb{P}(X_{t+1} = x \,|\, X_t),$$

▷ A Markov chain is defined by its **transition probability** p

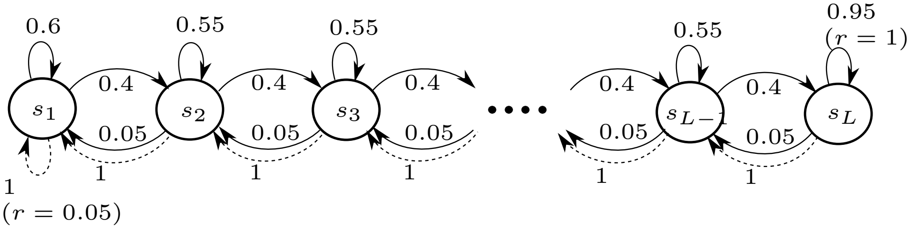
$$p(y|x) = \mathbb{P}(X_{t+1} = y|X_t = x).$$

▷ A stationary policy π induces a Markov chain in an MDP, with transition probability $p(s'|s) = \mathbf{p}(s, \pi(s))(s')$.

The formulation makes the reward and the state process be **Markov processes** with respect to the agent's actions and states:

$$\begin{aligned}
\mathbb{P}\Big(S_{t+1}\Big|S_t, A_t, R_t, \dots, S_1, A_1, R_1, S_0\Big) &= \mathbb{P}\Big(S_{t+1}\Big|S_t, A_t\Big)\\
\mathbb{P}\Big(R_{t+1}\Big|S_t, A_t, R_t, \dots, S_1, A_1, R_1, S_0\Big) &= \mathbb{P}\Big(R_{t+1}\Big|S_t, A_t\Big)
\end{aligned}$$

The action process is typically not Markov. However, the goal of a learner is typically to find a **stationary policy** such that $A_t \sim \pi(S_t) \in \mathcal{P}(\mathcal{A}_{S_t})$. In such case the knowledge of S_t is enough to determine the future of the system.

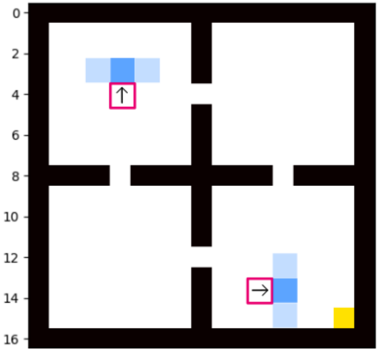


Actions: $\{\leftarrow, \rightarrow\}$

Rewards: 0 everywhere, except $\mathbf{r}(s_6, \rightarrow) = \delta_1$ and $\mathbf{r}(s_1, \leftarrow) = \delta_{0.05}$.

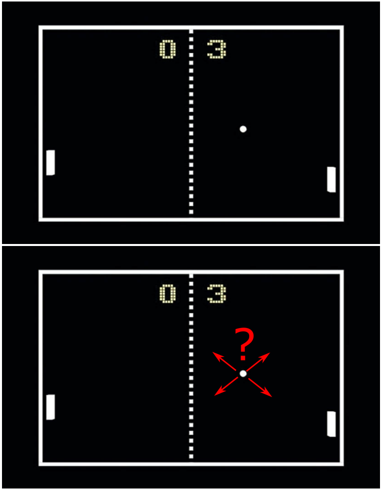
Transitions: $\mathbf{p}(s_\ell, \rightarrow) : s_{\ell+1} \mapsto 0.4, s_\ell \mapsto 0.55, s_{\ell-1} \mapsto 0.05$; $\mathbf{p}(s_\ell, \leftarrow) = \delta_{s_{\ell-1}}$.

Simple but flexible: L states, probabilities, "chain structure".



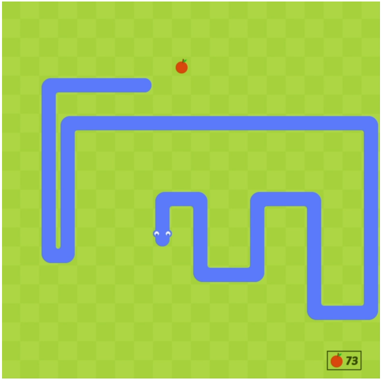
Actions: {⬅, ⬇, ➡, ⬆}
Rewards: 1 when in goal state, 0 otherwise.
Transitions: "Sloppy". Reset to random initial state after reaching goal (yellow) state. (See also "Frozen-lake")

- **Not** everything is an MDP: Careful how the state is defined.



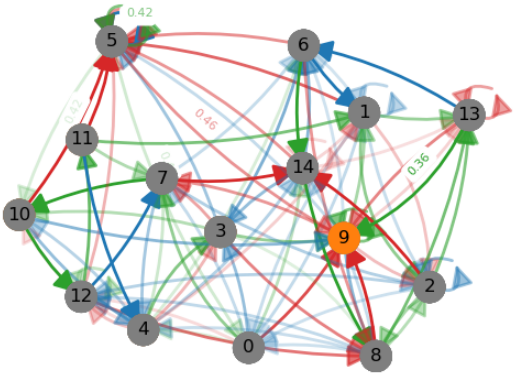
If s =screen pixels only, $P(s'|s)$ is **not** the same depending on past positions before current screen! Direction of ball?

- **Not** everything is an MDP: Self-avoiding paths.



Need to remember all previously visited positions.

"**G**eneralized **A**verage **R**eward **N**on-stationary **E**nvironment **T**est-bench"

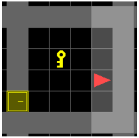
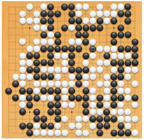


Transitions are randomly generated (random support, mass).

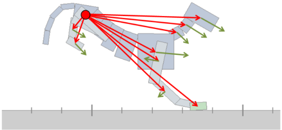
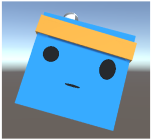
Rewards are randomly generated (e.g. truncated Gaussians with random mean) at random locations.

Parameters govern number of sates, actions, sparsity of support, minimal non-zero probability mass, sparsity of reward locations, etc.

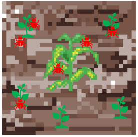
- Rather well-known dynamics, scarce rewards:

- Physics-based dynamics (white box), continuous state-action space.

- Real-world oriented, stochastic/black-box environment.

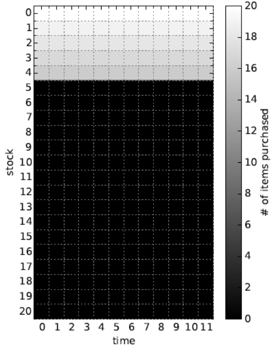



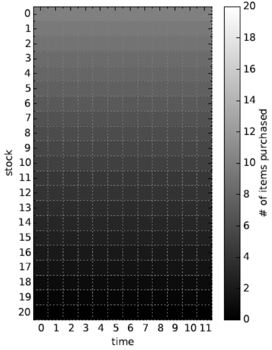
▷ At each month t, a store contains x_t **items** of a specific goods and the demand for that goods is D_t. At the end of each month the manager of the store can **order** a_t more items from his supplier. Furthermore we know that:

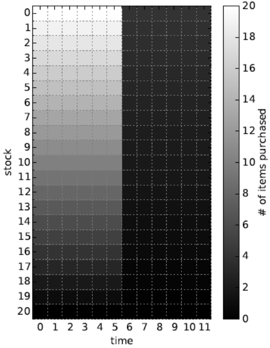
- The **cost** of maintaining an inventory of x is $h(x)$.
- The **cost** to order a items is $C(a)$.
- The **income** for selling q items is $f(q)$.
- If the demand D is bigger than the available inventory x, customers that cannot be served leave.
- The **value of the remaining inventory** at the end of the year is $g(x)$.
- **Constraint**: the store has a maximum capacity M.

- **State space**: $x \in \mathcal{X} = \{0, 1, \ldots, M\}$.
- **Action space**: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at statex, $a \in A(x) = \{0, 1, \ldots, M - x\}$.
- **Dynamics**: $x_{t+1} = [x_t + a_t - D_t]^+$, where D_t is the demand.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is **stochastic and time-independent**. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.
- **Reward**: $r_t = -C(a_t) - h(x_t + a_t) + f([x_t + a_t - x_{t+1}]^+)$.

Examples of policies: $\pi_t(x) =$







$(M-x)\mathbb{I}\{x<M/4\}$ $\quad$ $\max\{(M-x)/2-x;0\}$ $\quad$ $\begin{cases} M-x & \text{if } t<6 \\ \lfloor (M-x)/5 \rfloor & \text{otherwise} \end{cases}$

Consider a finite state space $\mathcal{S}$ with cardinality S, and a policy π.

- We denote $m_\pi \in \mathbb{R}^S$ the vector with components $m_\pi(s) = \mathbb{E}_{A\sim\pi}[\mathbf{m}(s,A)]$.
- We denote P_π the $S \times S$ matrix with components $P_\pi(s,s') = \mathbb{E}_{A\sim\pi}[\mathbf{p}(s,a)(s')]$.

For convenience, we also write $P_a(s'|s)$, or $P(s'|s,a)$ in lieu of $\mathbf{p}(s,a)(s')$.

A **policy** π is evaluated by the sum of rewards it enables to accumulate.

▷ The **T-horizon** **value function** of policy π in MDP M is:

$$V^\pi_T(s) = \mathbb{E}\Big[\sum_{t=1}^{T} R_t \Big| S_1 = s\Big] = \sum_{t=1}^{T}(P_\pi^{t-1} m_\pi)(s)\,.$$

Variant: T is first hitting time to reach some (termination) state.

▷ The **γ-discounted** **value function** of policy π in MDP M is:

$$V^\pi_\gamma(s) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \Big| S_1 = s\Big] = \sum_{t=1}^{\infty}((\gamma P_\pi)^{t-1} m_\pi)(s)\,.$$

For normalization purpose, we let $\overline{V}^\pi_T = \frac{1}{T} V^\pi_T$ and $\overline{V}^\pi_\gamma = (1-\gamma) V^\pi_\gamma$.

▷ The **average-reward** **value function** of policy π (when defined) is:

$$V^\pi(s) = \lim_{T\to\infty} \overline{V}^\pi_T(s)\,.$$

Decision-maker goal

Typical goal of an agent can be:

- **Find** a policy with **highest value** (no matter what rewards the agent receives).
- **Accumulate** maximum of rewards while interacting (maximize its own value).

In both case, we compare to the best possible value achievable by a policy.

Non-trivial problems to solve (given $\mathbf{m}, \mathbf{p}$):

- Given a policy π, how to **Evaluate** its value ?
- How to compute an **optimal policy** ?

Optimal Value Function

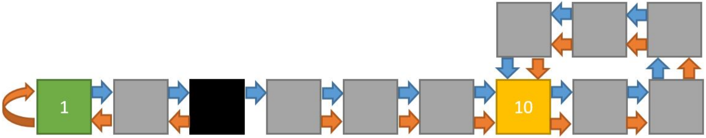
- An **optimal policy** $\pi^\star$ satisfies

$$\pi^\star \in \arg\max_{\pi \in \Pi} V^\pi$$

in all the states $s \in \mathcal{S}$, where Π is some policy set of interest.

The corresponding value function is the **optimal value function** $V^\star = \max_{\pi \in \Pi} V^\pi$

- An MDP may admit **more than one** optimal policy
- $\pi^\star$ achieves the largest possible value function in **every** state
- there always exists an optimal **deterministic** policy
- except for problems with a finite horizon, there always exists an optimal **stationary** policy



Actions: {←, →}, deterministic transitions.

The set of **optimal sequences** of actions can be made explicit for each horizon T:

T	1	2	3	$4,\dots,7$	8	9	$10,\dots,13$	14	
$\star$	$\mathcal{A}$	a^2	a^3	$b\mathcal{A}^{T-1}$	$a^2b^3\mathcal{A}^{T-5}$	$a^3b^3\mathcal{A}^{T-6}$	$b\mathcal{A}^{T-1}$	$a^2b^3\mathcal{A}^{T-5}$	$a^3 b$
V	0	1	2	10	11	12	20	21	

Here **no** sequence (hence policy) can be made **simultaneously optimal** for all time T (or even for all large enough T).

Table of contents

The expected return of **policy** π starting in state s at time t_1 is:

$$V^{\pi}_{t_1:T}(s) = \mathbb{E}\Big[\sum_{t=t_1}^{T} R_t \Big| S_{t_1} = s\Big] = \sum_{t=t_1}^{T} (P_{\pi}^{t-t_1} m_{\pi})(s)\,.$$

How can we **evaluate** $V^{\pi}_{1:T}(s)$ for some s?

- Estimate by **simulation** and Monte-Carlo: **approximate**.
- Develop **tree** of all possible realizations: $O(e^T)$ **many**.

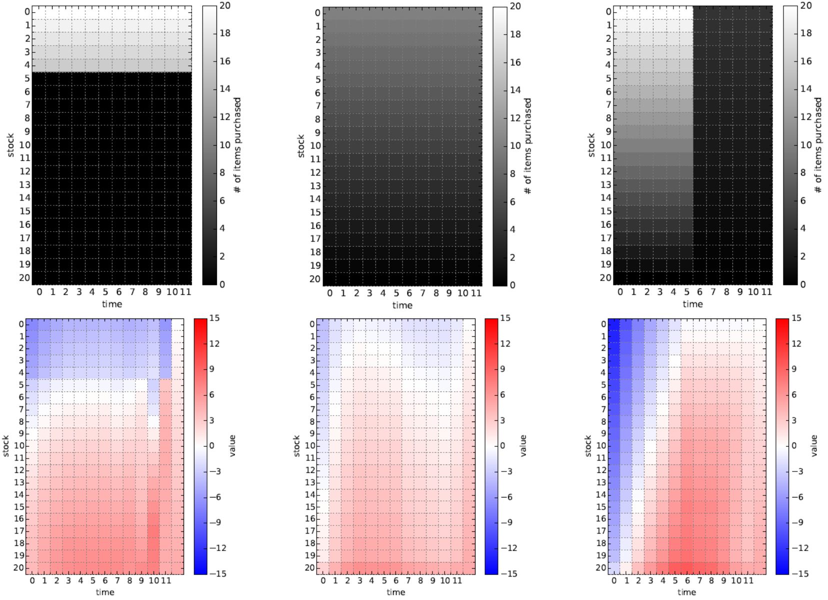
- The computation of $V^\pi_{t_1:T}$ can be done with the knowledge of $V^\pi_{\mathbf{t_1+1}:T}$:

$$
\begin{aligned}
V^\pi_{t_1:T}(s) &= \sum_{t=t_1}^{T} (P_\pi^{t-t_1} m_\pi)(s) \\
&= m_\pi(s) + \sum_{t=t_1+1}^{T} P_\pi (P_\pi^{t-t_1-1} m_\pi)(s) \\
&= m_\pi(s) + \sum_{s'\in\mathcal{S}} P_\pi(s,s') \sum_{t=t_1+1}^{T} (P_\pi^{t-t_1-1} m_\pi)(s') \\
&= m_\pi(s) + \sum_{s'\in\mathcal{S}} P_\pi(s,s') V^\pi_{\mathbf{t_1+1}:T}(s')
\end{aligned}
$$

Starting from $V^\pi_{T:T} = m_\pi$, we can then compute $V^\pi_{T-1:T}, V^\pi_{T-2:T}, \dots$ using in total $O(S^2)$ each time: $O(S^2 T)$ **computations for** $V^\pi_{1:T}$!

Dynamic Programming is a method for solving complex problem by breaking it down into a collection of simpler **sub-problems**.

Value of policies for retail management



Finite-horizon optimal value and policy

- Optimal value function and optimal policy:

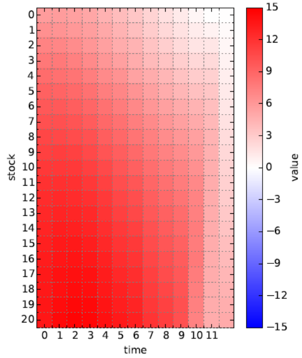
$$V_\star(s) = \max_{\pi=(\pi_1,\dots,\pi_T)} V_{\pi,1:T}(s)\,,$$

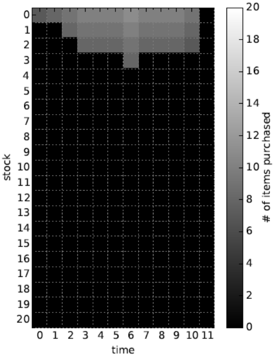
with optimal policy being a maximizer $\pi^\star = (\pi^\star_1,\dots,\pi^\star_T)$.

- **Naive** optimization : $\mathcal{O}(|\mathcal{S}|^{|\mathcal{A}|T})$
- However, we can show that $V_{\star,t:T} : s \mapsto \max_{\pi=(\pi_t,\dots,\pi_T)} V_{\pi,t:T}(s)$ satisfies

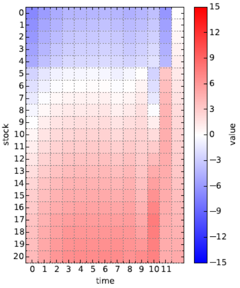
$$V_{\star,t:T}(s) = \max_{\mathbf{a}\in\mathcal{A}}\Big(m_{\mathbf{a}}(s) + \sum_{s'\in\mathcal{S}} P_{\mathbf{a}}(s,s')V^\pi_{\star,\mathbf{t_1+1}:T}(s')\Big)\,.$$

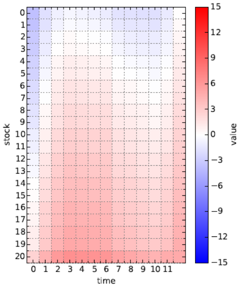
- Hence, we can compute $V_{\star,1:T}$ with $O(S^2AT)$ many computations ! (Further $\pi^\star_t(s)$ maximizes the r.h.s.)

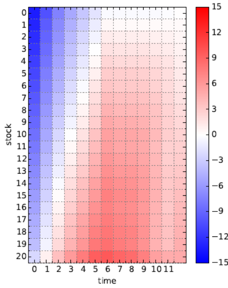




$V_\star \quad \pi_\star$







$V_{\pi^{(1)}}, V_{\pi^{(2)}}, V_{\pi^{(3)}}$

▷ When horizon T given, in order to compute the **T-horizon** **value function**

$$V_T^\pi(s) = \mathbb{E}\Big[\sum_{t=1}^{T} R_t \Big| S_1 = s\Big] = \sum_{t=1}^{T}(P_\pi^{t-1} m_\pi)(s)\,,$$

use **Dynamic Programming** for each policy π

$$V^\pi_{t_1:T}(s) \quad = \quad m_\pi(s) + \sum_{s'\in\mathcal{S}} P_\pi(s,s')V^\pi_{\mathbf{t_1+1}:T}(s')\,.$$

or to compute an **optimal** value and policy:

$$V_{\star,t:T}(s) = \max_{\mathbf{a}\in\mathcal{A}}\Big(m_{\mathbf{a}}(s) + \sum_{s'\in\mathcal{S}} P_{\mathbf{a}}(s,s')V^\pi_{\star,\mathbf{t_1+1}:T}(s')\Big)\,.$$

▷ $O(S^2AT)$ instead of $O(S^{AT})$ computations.

Table of contents

- Consider **infinite** horizon.

$$\max_{\pi} \; V_\gamma^\pi(s_0) =$$

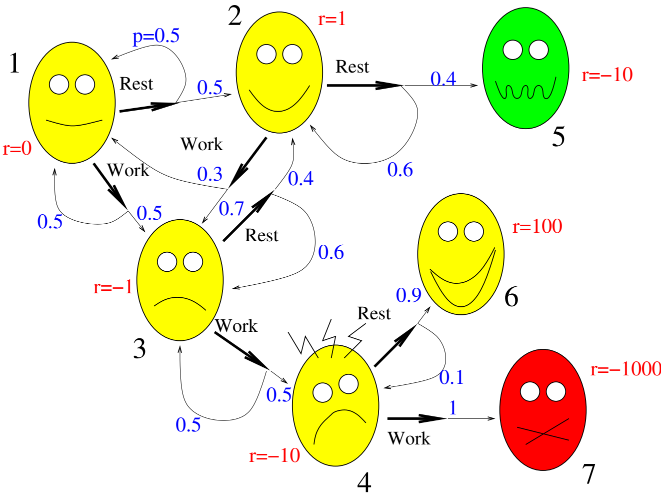
$$\max_{\pi} \; \mathbb{E}\big[\mathbf{m}(s_0,\pi(s_0)) + \gamma\mathbf{m}(s_1,\pi(s_1)) + \gamma^2\mathbf{m}(s_2,\pi(s_2)) + \dots\big]$$

$$\Downarrow$$

very challenging (we should try as many as $|A|^{|S|}$ policies!)

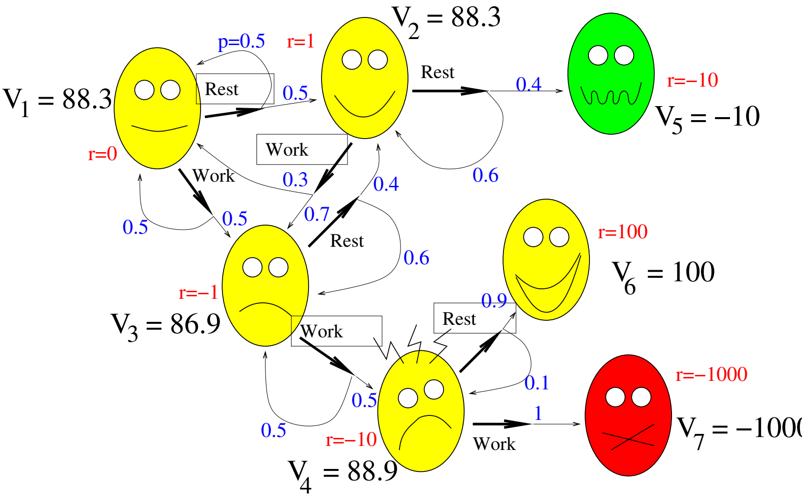
$$\Downarrow$$

we need to leverage the **structure** of the MDP
to **simplify** the optimization problem



States: x_5, x_6, x_7 are terminal.

Setting: infinite horizon with terminal states.



$V_1 = 88.3$
$V_2 = 88.3$
$V_3 = 86.9$
$V_4 = 88.9$
$V_5 = -10$
$V_6 = 100$
$V_7 = -1000$
p=0.5
r=1
r=0
r=−1
r=−10
r=100
r=−1000
Rest
Work
0.5
0.4
0.6
0.3
0.7
0.9
0.1
1

- $V_5 = -10$, $V_6 = 100$, $V_7 = -1000$ is immediate.
- Computing V_4:

$$\begin{cases} V_6 &= 100 \\ V_4 &= -10 + (0.9V_6 + 0.1V_4) \end{cases} \Rightarrow V_4 = \frac{-10 + 0.9V_6}{0.9} = 88.8$$

- Computing V_3: *no need* to consider all possible trajectories

$$\begin{cases} V_4 &= 88.8 \\ V_3 &= -1 + (0.5V_4 + 0.5V_3) \end{cases} \Rightarrow V_3 = \frac{-1 + 0.5V_4}{0.5} = 86.8$$

- And so on for V_1, V_2.

▷ The value function and optimal value function satisfy the following **Bellman fixed-point equations**:

$$\forall \pi \text{ stationary}, \quad \mathbf{V}^{\pi}(s) = \mathbf{m}(s, \pi(s)) + \gamma \sum_{s'} \mathbf{p}(s, \pi(s))(s')\mathbf{V}^{\pi}(s')$$

$$\mathbf{V}^{\star}(s) = \max_{a \in \mathcal{A}} \Big(\mathbf{m}(s, a) + \gamma \sum_{s'} \mathbf{p}(s, a)(s')\mathbf{V}^{\star}(s') \Big),$$

▷ That is $V^{\pi} = \mathcal{T}_{\pi}[V^{\pi}]$ and $V^{\star} = \mathcal{T}[V^{\star}]$, where we introduced the operators:

$$\begin{array}{rrcl} \text{(Bellman operator)} & \mathcal{T}_{\pi}[v] & = & m_{\pi} + \gamma P_{\pi} v \\ \text{(Bellman optimal operator)} & \mathcal{T}[v] & = & \max_a m_a + \gamma P_a v \end{array}$$

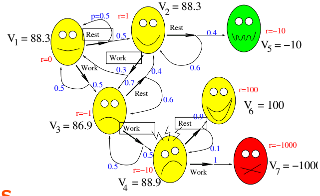
$$
\begin{aligned}
V^\pi(x) &= \mathbb{E}_\pi\left[\sum_{t\geqslant 0} \gamma^t R(x_t, \pi(x_t)) \,|\, x_0 = x\right] \\
&= \mathbf{m}(x, \pi(x)) + \mathbb{E}_\pi\left[\sum_{t\geqslant 1} \gamma^t R(x_t, \pi(x_t)) \,|\, x_0 = x\right] \\
&= \mathbf{m}(x, \pi(x)) \\
&\qquad + \gamma \sum_y \mathbb{P}(x_1 = y \,|\, x_0 = x; \pi(x_0)) \mathbb{E}_\pi\left[\sum_{t\geqslant 1} \gamma^{t-1} R(x_t, \pi(x_t)) \,|\, x_1 = y\right] \\
&= \mathbf{m}(x, \pi(x)) + \gamma \sum_y \mathbf{p}(x, \pi(x))(y) V^\pi(y). \qquad \blacksquare
\end{aligned}
$$

Proceed similarly for $V^\star$.

The **Bellman fixed-point equation**

$$\mathbf{V}^\pi(x) = m_\pi(x) + \gamma \sum_y p_\pi(y|x)\mathbf{V}^\pi(y).$$

is a **linear** system of equations with N unknowns and N linear constraints.



$$\mathbf{V}^{\pi}(x) = m_{\pi}(x) + \gamma \sum_y p_{\pi}(y|x)\mathbf{V}^{\pi}(y)$$

System of equations

$$\left\{\begin{array}{ll} V_1 & = 0 + 0.5V_1 + 0.5V_2 \\ V_2 & = 1 + 0.3V_1 + 0.7V_3 \\ V_3 & = -1 + 0.5V_4 + 0.5V_3 \\ V_4 & = -10 + 0.9V_6 + 0.1V_4 \\ V_5 & = -10 \\ V_6 & = 100 \\ V_7 & = -1000 \end{array}\right. \quad \Rightarrow$$

$$(V, m \in \mathbb{R}^7,\ P \in \mathbb{R}^{7\times 7})$$

$$V = m + PV$$

$$\Downarrow$$

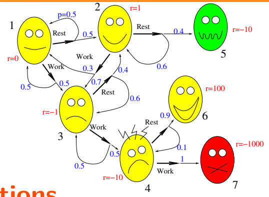
$$V = (I - P)^{-1} m$$

The **optimal Bellman fixed-point** equation

$$\mathbf{V}^*(x) = \max_{a\in A}\big[r(x,a) + \gamma \sum_y p(y|x,a)\mathbf{V}^*(y)\big].$$

is a (highly) **non-linear** system of equations with N unknowns and N non-linear constraints (i.e., the max operator).

$$\mathbf{V}^*(x) = \max_{\mathbf{a}\in\mathbf{A}} \Big[r(x,a) + \gamma \textstyle\sum_y p(y|x,a)\mathbf{V}^*(y) \Big]$$



System of equations

$$\begin{cases}
\mathbf{V}_1 & = \max\big\{0 + 0.5\mathbf{V}_1 + 0.5\mathbf{V}_2,\ \ 0 + 0.5\mathbf{V}_1 + 0.5\mathbf{V}_3\big\}\\
\mathbf{V}_2 & = \max\big\{1 + 0.4\mathbf{V}_5 + 0.6\mathbf{V}_2,\ \ 1 + 0.3\mathbf{V}_1 + 0.7\mathbf{V}_3\big\}\\
\mathbf{V}_3 & = \max\big\{-1 + 0.4\mathbf{V}_2 + 0.6\mathbf{V}_3,\ \ -1 + 0.5\mathbf{V}_4 + 0.5\mathbf{V}_3\big\}\\
\mathbf{V}_4 & = \max\big\{-10 + 0.9\mathbf{V}_6 + 0.1\mathbf{V}_4,\ \ -10 + \mathbf{V}_7\big\}\\
\mathbf{V}_5 & = -10\\
\mathbf{V}_6 & = 100\\
\mathbf{V}_7 & = -1000
\end{cases}$$

$\Rightarrow$ too complicated, we need to find an **alternative** solution.

- ▷ Solution 1: Linear programming.
- ▷ Solution 2: Bellman iteration.

Algorithm 1 Linear programming

1: Let $v_\star$ be the solution to

$$\min_{v \in \mathbb{R}^S} \sum_{s \in \mathcal{S}} v(s) \tag{1}$$

$$\text{subject to } v(s) \geqslant \mathbf{m}(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v(s'), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \tag{2}$$

2: **return** the policy π_* defined as

$$\pi_\star(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(\mathbf{m}(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\star(s') \right) . \tag{3}$$

- Discounted **Bellman operators**:

$$\begin{array}{lrcl} \text{(Bellman step)} & \mathcal{T}_\pi[v] & = & m_\pi + \gamma P_\pi v \\ \text{(Bellman optimal)} & \mathcal{T}[v] & = & \max_a m_a + \gamma P_a v \\ \text{(Bellman greedy)} & \mathcal{G}[v] & = & \operatorname{Argmax}_a m_a + \gamma P_a v \end{array}$$

- Bellman fixed points equations are

$$V_\pi = \mathcal{T}_\pi[V_\pi], \quad V_\star = \mathcal{T}[V_\star]$$

- Linear programming is:

$$\begin{array}{rcl} v_\star & = & \operatorname{argmin}_{v\in\mathbb{R}^S} \sum_{s\in\mathcal{S}} v(s) \text{ s.t. } \forall s, v(s) \geqslant \mathcal{T}[v](s), \\ \pi_\star & = & \mathcal{G}[v_\star]. \end{array}$$

- [**Crucial !**] γ-**contraction** property (in $\|\cdot\|_\infty$ norm).

$$\|\mathcal{T}_\pi[v]-\mathcal{T}_\pi[w]\|_\infty \leqslant \gamma\|v-w\|_\infty < \|v-w\|_\infty\,.$$
$$\|\mathcal{T}[v]-\mathcal{T}[w]\|_\infty \leqslant \gamma\|v-w\|_\infty < \|v-w\|_\infty\,.$$

▷ [**Crucial !**] γ-**contraction** property (in $\|\cdot\|_\infty$ norm).

$$\|\mathcal{T}_\pi[v] - \mathcal{T}_\pi[w]\|_\infty \leqslant \gamma\|v - w\|_\infty < \|v - w\|_\infty \,.$$
$$\|\mathcal{T}[v] - \mathcal{T}[w]\|_\infty \leqslant \gamma\|v - w\|_\infty < \|v - w\|_\infty \,.$$

This entails (Banach fixed point theorem):

- Existence of a (unique) **fixed** point: V^π of $\mathcal{T}_\pi$, $V^\star$ of $\mathcal{T}$.
- Possibility to resort to an **iterative** algorithm: $\|V^\star - \mathcal{T}^k\mathbf{0}\|_\infty \leqslant \gamma^k\|V^\star\|_\infty \leqslant \varepsilon$ for $k \geqslant \frac{\log(\|V^\star\|_\infty/\varepsilon)}{\log(1/\gamma)}$ where $\|V^\star\|_\infty \leqslant \frac{\max_{s,a}\mathbf{m}(s,a)}{1-\gamma}$.

Other properties

▷ **Monotony**: for any $W_1, W_2 \in \mathbb{R}^N$, if $W_1 \leqslant W_2$ component-wise, then

$$\mathcal{T}^\pi W_1 \leqslant \mathcal{T}^\pi W_2, \qquad \mathcal{T} W_1 \leqslant \mathcal{T} W_2.$$

▷ **Offset**: for any scalar $c \in \Re$,

$$\mathcal{T}^\pi(W + cI_N) = \mathcal{T}^\pi W + \gamma \mathbf{c} I_N, \qquad \mathcal{T}(W + cI_N) = \mathcal{T} W + \gamma \mathbf{c} I_N,$$

The contraction property (3) holds since for any $x \in X$ we have

$$
\begin{aligned}
&|\mathcal{T}W_1(x) - \mathcal{T}W_2(x)| \\
&\quad = \left| \max_a \left[m_a(x) + \gamma \sum_y p(y|x,a) W_1(y) \right] - \max_{a'} \left[m_{a'}(x) + \gamma \sum_y p(y|x,a') W_2(y) \right] \right| \\
&\quad \overset{(a)}{\leqslant} \max_a \left| \left[m_a(x) + \gamma \sum_y p(y|x,a) W_1(y) \right] - \left[m_a(x) + \gamma \sum_y p(y|x,a) W_2(y) \right] \right| \\
&\quad = \gamma \max_a \sum_y p(y|x,a) |W_1(y) - W_2(y)| \\
&\quad \leqslant \gamma ||W_1 - W_2||_\infty \max_a \sum_y p(y|x,a) = \gamma ||W_1 - W_2||_\infty,
\end{aligned}
$$

where in (a) we used $\max_a f(a) - \max_{a'} g(a') \leqslant \max_a (f(a) - g(a))$. ■

▷ Applying the Bellman operator at each step plus using $\pi_{n+1} \in \mathcal{G}[v_n]$ yields:

$$(\textbf{Value Iteration}) \qquad v_{n+1} = \mathcal{T}[v_n] = \mathcal{T}_{\pi_n}[v_n]$$

Hence:

$$v_{n+1} = \mathcal{T}_{\pi_n}\mathcal{T}_{\pi_{n-1}}\dots,\mathcal{T}_{\pi_0}[v_0]$$

▷ **Convergence**: For any v_0, if $v_{n+1} = \mathcal{T}[v_n]$ then $v_n \to_{||\cdot||} V_\star$ with a **geometric convergence rate**:

$$||v_n - V_\star|| \leqslant \gamma^n ||v_0 - V_\star||.$$

▷ Time complexity $O(n|S|^2|A|)$ after n steps, Space complexity: dynamics $O(|S|^2|A|)$, reward $O(|S||A|)$, value function and optimal policy $O(|S|)$.

- Applying the Bellman operator at each step plus using $\pi_{n+1} \in \mathcal{G}[v_n]$ yields:

$$(\textbf{Value Iteration}) \qquad v_{n+1} = \mathcal{T}[v_n] = \mathcal{T}_{\pi_n}[v_n]$$

Other Bellman iteration algorithms

▷ Applying the Bellman operator at each step plus using $\pi_{n+1} \in \mathcal{G}[v_n]$ yields:

$$(\textbf{Value Iteration}) \qquad v_{n+1} = \mathcal{T}[v_n] = \mathcal{T}_{\pi_n}[v_n]$$

▷ Q: Why not iterating Bellman operator several times (Still using $\pi_{n+1} \in \mathcal{G}[v_n]$)?

$$(\textbf{Policy Iteration}) \qquad v_{n+1} = \mathcal{T}_{\pi_n}^{\infty}[v_n] = (1-\gamma P_{\pi_n})^{-1}\mu_{\pi_n}$$

$$(m\text{-PI, Puterman\&Shin 78}) \qquad v_{n+1} = \mathcal{T}_{\pi_n}^{m+1}[v_n], m \in \mathbb{N}$$

$$(\lambda\text{-PI, Ioffe\&Bertsekas 96}) \qquad v_{n+1} = (1-\lambda)\sum_{m=0}^{\infty} \lambda^m \mathcal{T}_{\pi_n}^{m+1}[v_n], \quad \lambda \in [0,1)$$

$$(\text{Opt-PI, Thiéry\&Scherrer 09}) \qquad v_{n+1} = \sum_{m=0}^{\infty} \lambda_m \mathcal{T}_{\pi_n}^{m+1}[v_n], \sum_m \lambda_m = 1, \lambda_m > 0$$

▷ Thanks to the γ-**contraction** property, all these algorithms **converge asymptotically** to an **optimal** value-policy pair $V_\star, \pi_\star$. (Theorem Puterman&Shin 78, Ioffe&Bertsekas 96, Thiéry&Scherrer 09).

▷ Especially interesting in practice when $\gamma \simeq 1$.

- Start with some policy π_1, $n = 1$.
- Compute $v_{n+1} = \mathcal{T}_{\pi_n}^{\infty}[v_n] = (1 - \gamma P_{\pi_n})^{-1}\mu_{\pi_n}$ [Policy Evaluation]
- Let $\pi_{n+1} \in \mathcal{G}[v_n]$ [Policy Improvement]
- Stop if $v_{n+1} = v_n$, else $n = n + 1$

Theorem

PI generates a sequence of policies with **non-decreasing** values: $V_{\pi_{n+1}} \geqslant V_{\pi_n}$. Then the MDP is finite, **convergence** occurs in a **finite number** of iterations.

- Monotony:

$$\begin{aligned} V_{\pi_{n+1}} &= (1-\gamma P_{\pi_{n+1}})^{-1}(\mu_{\pi_{n+1}} + \gamma P_{\pi_{n+1}} V_{\pi_n} - \gamma P_{\pi_n} V_{\pi_n}) \\ &= (1-\gamma P_{\pi_{n+1}})^{-1}\mathcal{T}_{\pi_{n+1}} V_{\pi_n} - (1-\gamma P_{\pi_{n+1}})^{-1} V_{\pi_n} + V_{\pi_n} \\ &= (1-\gamma P_{\pi_{n+1}})^{-1}\left[\mathcal{T}_{\pi_{n+1}} V_{\pi_n} - V_{\pi_n}\right] + V_{\pi_n} \\ &= (1-\gamma P_{\pi_{n+1}})^{-1}\left[\mathcal{T} V_{\pi_n} - \mathcal{T}_{\pi_n} V_{\pi_n}\right] + V_{\pi_n} \\ &\geqslant V_{\pi_n}\,. \end{aligned}$$

- Optimality: If $V_{\pi_{n+1}} = V_{\pi_n}$, then $V_{\pi_n} = \mathcal{T}_{\pi_{n+1}} V_{\pi_{n+1}} = \mathcal{T}_{\pi_{n+1}} V_{\pi_n} = \mathcal{T} V_{\pi_n}$.

Hence $V_{\pi_n} = V_\star$, by uniqueness of the fixed point.

- Termination occurs because a finite MDP has finitely many policies.

- **Direct computation.** For any policy π compute

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} r^{\pi}.$$

Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

- **Direct computation.** For any policy π compute

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi.$$

Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

- **Iterative policy evaluation.** For any policy π

$$\lim_{n\to\infty} \mathcal{T}^\pi V_0 = V^\pi.$$

Complexity: An ε-approximation of V^π requires $O(N^2 \frac{\log 1/\varepsilon}{\log 1/\gamma})$ steps.

- **Direct computation.** For any policy π compute

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi.$$

  Complexity: $O(N^3)$ (improvable to $O(N^{2.807})$).

- **Iterative policy evaluation.** For any policy π

$$\lim_{n\to\infty} \mathcal{T}^\pi V_0 = V^\pi.$$

  Complexity: An ε-approximation of V^π requires $O(N^2 \frac{\log 1/\varepsilon}{\log 1/\gamma})$ steps.

- **Monte-Carlo simulation.** In each state s, simulate n trajectories $((s_t^i)_{t\geqslant 0,})_{1\leqslant i\leqslant n}$ following policy π and compute

$$\widehat{V}^\pi(s) \simeq \frac{1}{n}\sum_{i=1}^n \sum_{t\geqslant 0} \gamma^t r(s_t^i, \pi(s_t^i)).$$

Complexity: In each state, the approximation error is $O(1/\sqrt{n})$.

▷ When horizon is **infinite**, in order to find a policy with maximal **γ-discounted value function**

$$V_\gamma^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \middle| S_1 = s\right] = \sum_{t=1}^{\infty} ((\gamma P_\pi)^{t-1} m_\pi)(s) \,,$$

▷ Value Iteration:

- $v_{n+1} = \mathcal{T}_{\pi_n}[v_n]$, $\pi_{n+1} \in \mathcal{G}[v_n]$
- Each iteration has complexity $O(|S|^2|A|)$
- Asymptotic convergence

▷ Policy iteration :

- $v_{n+1} = \mathcal{T}_{\pi_n}^{\infty}[v_n]$, $\pi_{n+1} \in \mathcal{G}[v_n]$
- Each iteration has complexity $O(|S|^2|A|) + O(|S|^3)$
- Convergence in finite time: at most $O(\frac{|S||A|}{1-\gamma} \log(\frac{1}{1-\gamma}))$ iterations (Ye, 2010, Hansen 2011, Scherrer 2013).

▷ The **Quality function** of policy π at (s,a) is the value of first taking action a, then policy π:

$$q_\pi(s,a) = m(s,a) + \gamma(P_a v_\pi)(s).$$

▷ Bellman **q-operators** and **Fixed point** equations :

$$\begin{aligned}
\text{(Bellman step)} \quad \mathcal{T}_\pi[q] &= (s,a) \mapsto \mathbf{m}(s,a) + \gamma \sum_{s'} P_a(s'|s) q(s', \pi(s'))\\
\text{(Bellman optimal)} \quad \mathcal{T}[q] &= (s,a) \mapsto \mathbf{m}(s,a) + \gamma \sum_{s'} P_a(s'|s) \max_a q(s', a)\\
\text{(Bellman greedy)} \quad \mathcal{G}[q] &= s \mapsto \operatorname{Argmax}_a q(s,a)
\end{aligned}$$

$$q_\pi = \mathcal{T}_\pi[q_\pi], \quad q_\star = \mathcal{T}[q_\star], \quad \pi_\star = \mathcal{G}[q_\star]\,.$$

▷ **Relations** between V and q:

$$\begin{aligned}
V_\pi(s) &= q_\pi(s, \pi(s)), & q_\pi(s,a) &= \mathcal{T}_a[V_\pi](s)\\
V_\star(s) &= \max_a q_\star(s,a), & q_\star(s,a) &= \mathcal{T}_a[V_\star](s)\,.
\end{aligned}$$

- **Q-iteration**: compute $q_{n+1} = \mathcal{T}[q_n]$ until stopping criterion; return $\pi = \mathcal{G}[q_N]$

- Q-values are values in **augmented** problem $\mathcal{X} \times \mathcal{A}$.
- Space complexity: $O(|\mathcal{S}||\mathcal{A}|)$ instead of $O(|\mathcal{S}|)$.
- Time complexity: Computing $\mathcal{G}[q]$ is $O(|\mathcal{A}|)$ instead of $O(|\mathcal{S}|^2|\mathcal{A}|)$.
- No need to store π $(\operatorname{argmax} q)$.

- In VI or PI, we perform **full** updates of the value v_n, that is we update v_n **for all** x before moving to next update.
- Perform **local** updates instead:
  - Choose one $\mathbf{x_n}$ at step n
  - Compute $v_{n+1}(x_n) = \mathcal{V}[v_n](x_n)$.
- Using fancy scheduling of updates (**prioritized updates**) can drastically reduce number of iterations until convergence.

- VI, PI require to know P, R exactly (**model-based**).
- How to do **Policy evaluation** when P, R unknown?

Key observation:

V^π is an **expectation**.

- We can **sample** trajectories ("Roll-outs") $(s_{i,t}, r_{i,t})_{i,t}$ of interaction with π, where $r_{i,t} \sim \mathbf{r}(s_{i,t}, \pi(s_{i,t}))$ then do **"Monte-Carlo" estimate**:

$$V^\pi = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n \Big(\sum_{t=0}^\infty \gamma^t r_{i,t}\Big),$$

- We can update V^π estimate using **Incremental** updates
  - "Temporal difference" methods $TD(0)$, $TD(1)$, $TD(\lambda)$.

Let us explain this.

(Fancy variant: First-visit, Every-visit)

Idea: approximate an **expectation** by **incremental** updates

Idea: approximate an **expectation** by **incremental** updates

- Estimated value function **after trajectory** i generated with policy π

$$\big(\mathrm{TD}(1)\big) \qquad \widehat{V}_i^{\pi}(s_0) = (1-\alpha_i)\widehat{V}_{i-1}^{\pi}(s_0) + \alpha_i \sum_{t=0}^{T^{(i)}} \gamma^t r_{i,t}$$

Note: $\alpha_i = 1/i$ is an incremental version of the empirical mean

Idea: approximate an **expectation** by **incremental** updates

- Estimated value function **after trajectory** i generated with policy π

$$\big(\mathrm{TD}(1)\big) \qquad \widehat{V}^{\pi}_{i}(s_0) = (1-\alpha_i)\widehat{V}^{\pi}_{i-1}(s_0) + \alpha_i \sum_{t=0}^{T^{(i)}} \gamma^t r_{i,t}$$

Note: $\alpha_i = 1/i$ is an incremental version of the empirical mean

- Estimated value function **after transition** $\langle s_t, r_t, s_{t+1}\rangle$

$$\big(\mathrm{TD}(0)\big) \quad \widehat{V}^{\pi}(s_t) = (1-\alpha_i(s_t))\widehat{V}^{\pi}(s_t) + \alpha_i(s_t)\big(r_t + \gamma \widehat{V}^{\pi}(s_{t+1})\big)$$

$$= \widehat{V}^{\pi}(s_t) + \alpha_i(s_t)\big(\underbrace{r_t + \gamma\widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t)}_{\text{Temporal difference } \delta_t}\big)$$

Idea: approximate an **expectation** by **incremental** updates

- Estimated value function **after trajectory** i generated with policy π

$$(\mathrm{TD}(1)) \qquad \widehat{V}_i^{\pi}(s_0) = (1-\alpha_i)\widehat{V}_{i-1}^{\pi}(s_0) + \alpha_i \sum_{t=0}^{T^{(i)}} \gamma^t r_{i,t}$$

Note: $\alpha_i = 1/i$ is an incremental version of the empirical mean

- Estimated value function **after transition** $\langle s_t, r_t, s_{t+1}\rangle$

$$(\mathrm{TD}(0)) \qquad \widehat{V}^{\pi}(s_t) = (1-\alpha_i(s_t))\widehat{V}^{\pi}(s_t) + \alpha_i(s_t)(r_t + \gamma \widehat{V}^{\pi}(s_{t+1}))$$

$$= \widehat{V}^{\pi}(s_t) + \alpha_i(s_t)\underbrace{(r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t))}_{\text{Temporal difference } \delta_t}$$

- **Mix updates**: for any $\lambda \in [0,1]$

$$(\mathrm{TD}(\lambda)) \qquad \widehat{V}^{\pi}(s_t) = \widehat{V}^{\pi}(s_t) + \alpha_i(s_t)\sum_{t'=t}^{T}(\gamma\lambda)^{t'-t}\delta_s$$

Policy evaluation for given π:

- Sampling: $V^\pi = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n}(\sum_{t=0}^{\infty}\gamma^t r_{i,t})$, $r_{i,t} \sim \mathbf{r}(s_{i,t}, \pi(s_{i,t}))$ for trajectories $s_i = (s_{i,t})_t$, $i \in \mathbb{N}$ ("Roll-out", "Monte-Carlo" estimate).
  Fancy variants:
  - First-visit, Every-visit,
  - Incremental updates: "Temporal difference" methods $TD(0)$, $TD(1)$, $TD(\lambda)$.

Policy evaluation for given π:

- Sampling: $V^\pi = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n(\sum_{t=0}^\infty \gamma^t r_{i,t})$, $r_{i,t} \sim \mathbf{r}(s_{i,t}, \pi(s_{i,t}))$ for trajectories $s_i = (s_{i,t})_t$, $i \in \mathbb{N}$ ("Roll-out", "Monte-Carlo" estimate). Fancy variants:
  - First-visit, Every-visit,
  - Incremental updates: "Temporal difference" methods $TD(0)$, $TD(1)$, $TD(\lambda)$.

Policy optimization: How to optimize over π?

- **Q-learning**

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(s_t, a_t) \underbrace{\left[r_t + \gamma \max_{b\in\mathcal{A}} Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\right]}_{\text{Q-Temporal difference}}.$$

Policy evaluation for given π:

- Sampling: $V^\pi = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n (\sum_{t=0}^\infty \gamma^t r_{i,t})$, $r_{i,t} \sim \mathbf{r}(s_{i,t}, \pi(s_{i,t}))$ for trajectories $s_i = (s_{i,t})_t$, $i \in \mathbb{N}$ ("Roll-out", "Monte-Carlo" estimate).
  Fancy variants:
  - First-visit, Every-visit,
  - Incremental updates: "Temporal difference" methods $TD(0)$, $TD(1)$, $TD(\lambda)$.

Policy optimization: How to optimize over π?

- **Q-learning**

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(s_t, a_t) \underbrace{\left[r_t + \gamma \max_{b\in\mathcal{A}} Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\right]}_{\text{Q-Temporal difference}}.$$

- "SARSA" variant: $[r_t + \gamma Q_t(s_{t+1}, \mathbf{a_{t+1}}) - Q_t(s_t, a_t)]$

- This is a **model-free** stochastic approximation algorithm.

Theorem

If $\alpha(s_t, a_t) = \alpha_t$ satisfies $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$, and if state-action pairs are selected infinitely often, then Q-learning converges a.s. to $Q_\star$.

- Proof is sophisticated, using **stochastic approximation** (Robbins-Monro algorithm) plus **asynchronous** algorithm and **contraction**.
- Does not work with "V" value due to **order of** max **and** $\mathbb{E}$:

$$\mathcal{T}[q](s,a) = \mathbb{E}\Big[R + \gamma \sum_{s'} P(s'|s,a)\max_{a'} q(s',a')\Big]$$

$$\mathcal{T}[v](s) = \max_a \mathbb{E}\Big[R + \gamma \sum_{s'} P(s'|s,a) v(s')\Big]$$

(TD works with V-value because there is no max)

Take home messages

Take home messages

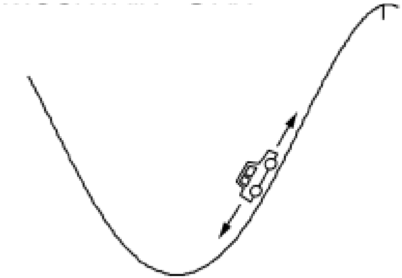
- MDPs helpful to model **dynamical** systems.
- State, Actions, Transition, Reward : (s,a,s',r)
- Sequential, repeated interactions (trajectories)
- Key property: **contraction** of Bellman operators allow **iterative** algorithms.
- DP, VI, PI: model-based strategies
- TD-learning, Q-learning: model-free strategies

We only talked about MDPs. But there are other models modeling other aspects:

- **Contextual** MDPs: Observe some external variable that affects dynamics (e.g. weather).
- **Piecewise-deterministic** MDPs: Restless setup with system continuously evolving with time.
- **Factored** MDPs: Compound states $s = (s_1, s_2)$, with factored dynamics

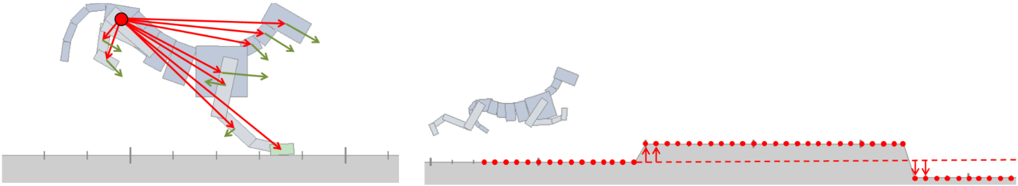
$$\mathbb{P}((s'_1, s'_2)|(s_1, s_2)) = \mathbb{P}(s'_1|s_1)\mathbb{P}(s'_2|s_2).$$

- **Relational** and **Object Oriented** MDPs: State uses occurrence of objects. Dynamics describe object-to-object interaction.
- **Causal** MDPs: State combines several variables, linked by causal graph.



States: Position $\times$ Speed of car (continuous).
Actions: Acceleration of car.
Transitions: Physics.
Rewards: 1 if reach top of mountain, 0 else.
Optimal policy: First go left to get momentum, then go right.

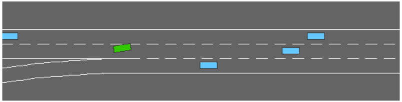
Continuous state-action space. For other classical tasks, check `https://gym.openai.com/envs/#classic_control`.



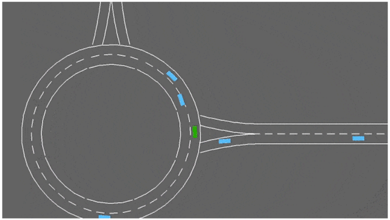
States: relative positions (red points) and velocities.
Actions: acceleration at each joint
Rewards: not falling (height of anchor point).

Example: High-way

Controlling an ego-vehicle in various road environments
`https://github.com/eleurent/highway-env`



The merge-v0 environment.



The roundabout-v0 environment.

Actions: acceleration, changing lane, etc.
Rewards: no collision, speed, etc.

Study safety and robustness (here with partially known behavior of other vehicles).
`http://edouardleurent.com/publication/phd-thesis/`

- How to even **store** V, Q, π when $S = \mathbb{R}^d$?
- DP, VI, PI require P, R: How do we represent them?
- **Function representation**:

$$\begin{aligned} \text{(Linear model)} \quad f_\theta(x) &= \sum_i \theta_i \varphi_i(x) \\ \text{(Neural network)} \quad f_\theta(x) &= \sum_{j_K} \theta_{K,j_K} \varphi_{K,j_K}\Big(\dots \varphi_{2,j_2}\Big(\sum_{j_1} \theta_{1,j_1}\varphi_{1,j_1}(x)\Big)\dots\Big) \end{aligned}$$

- This leads to

Approximate Dynamic Programming,
Approximate Value Iteration,
Approximate Policy Iteration,
Deep Q-network,
etc.

Noisy observations

- TD Q-learning perform mean updates from **sampled** trajectories.
- Ok for deterministic systems, but what happens in a **stochastic** system with stochastic transitions and rewards ?
- What is **error** between **estimated** value and **true** value?
- Can/Should/How do we correct for this error?

Exploration-Exploitation trade-off
Multi-armed bandits lecture

"The more applied you go, the stronger theory you need"

MERCI

odalricambrym.maillard@inria.fr
odalricambrymmaillard.wordpress.com