facebook
Artificial Intelligence Research

# How to model an RL problem: Markov Decision Processes

**Matteo Pirotta**
Facebook AI Research

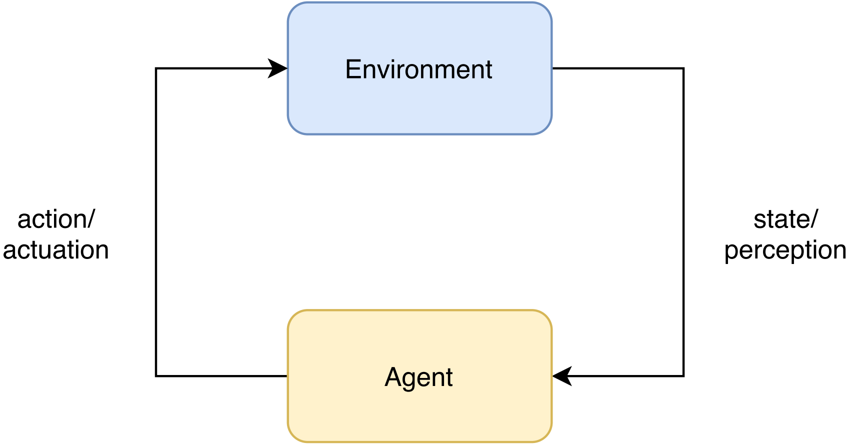
# Acknowledgments

Special thanks to Alessandro Lazaric for providing these slides from the RL class we teach in Paris.

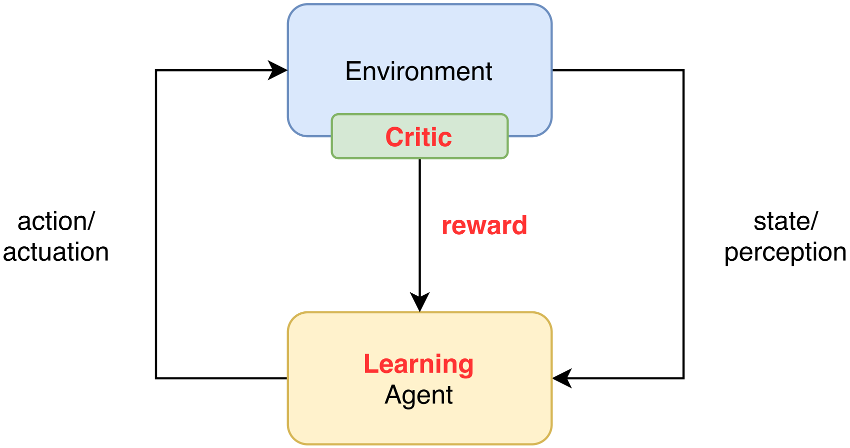
# Outline

# The Reinforcement Learning Model





# The Reinforcement Learning Model



# The RL interaction



In each *discrete* decision time $t = 1, 2, \ldots$, the learning agent

- selects an action $a_t$ based on the current state $s_t$ (or possibly all previous observations)
- observes a reward $r_t$
- moves to a new state $s_{t+1}$

# Markov Chains

**Definition (Markov chain)**

Let the *state space* $S$ be a bounded compact subset of the Euclidean space, the discrete-time dynamic system $(s_t)_{t\in\mathbb{N}} \in X$ is a Markov chain if it satisfies the *Markov property*

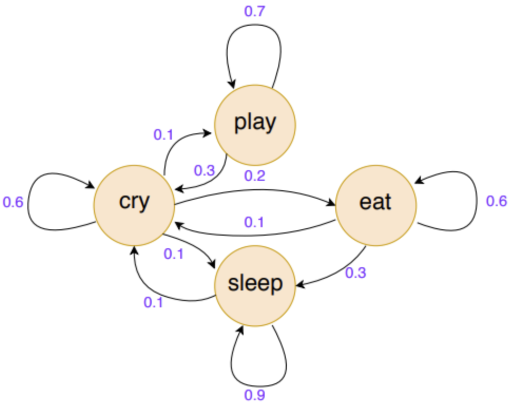
$$\mathbb{P}(s_{t+1} = s \,|\, s_t, s_{t-1}, \ldots, s_0) = \mathbb{P}(s_{t+1} = s \,|\, s_t),$$

Given an initial state $s_0 \in S$, a Markov chain is defined by the *transition probability* $p$

$$p(s'|s) = \mathbb{P}(s_{t+1} = s'|s_t = s).$$

# Markov Chain






$4$ states Markov chain

# Markov Decision Process



**Definition (Markov decision process [1, 4, 3, 6, 2])**

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- $S$ is the *state* space,
- $A$ is the *action* space,
- $p(s'|s, a)$ is the *transition probability* with

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

- $r(s, a, s')$ is the *reward* of transition $(s, a, s')$.

# Markov Decision Process



**Definition (Markov decision process [1, 4, 3, 6, 2])**

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- $S$ is the *state* space,
- $A$ is the *action* space,

finite

- $p(s'|s,a)$ is the *transition probability* with

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

- $r(s,a,s')$ is the *reward* of transition $(s,a,s')$. bounded and simplified to $r(s,a)$

# Markov Decision Process



**Definition (Markov decision process [1, 4, 3, 6, 2])**

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

$$\left.\begin{array}{l} \text{■ } S \text{ is the } \textit{state} \text{ space,} \\ \text{■ } A \text{ is the } \textit{action} \text{ space,} \end{array}\right\} \text{finite}$$

- $p(s'|s,a)$ is the *transition probability* with

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

- $r(s,a,s')$ is the *reward* of transition $(s,a,s')$. } bounded and simplified to $r(s,a)$

Reward may be stochastic

- $\nu(s,a)$ is the reward distribution for $(s,a)$ and

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

# Markov Decision Process

**Definition (Markov decision process [1, 4, 3, 6, 2])**

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- $S$ is the *state* space,
- $A$ is the *action* space,

finite

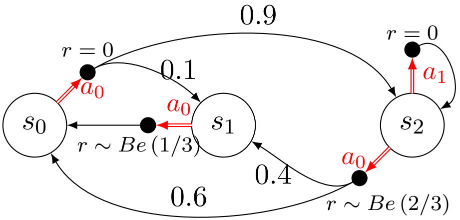
- $p(s'|s,a)$ is the *transition probability* with

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

- $r(s,a,s')$ is the *reward* of transition $(s,a,s')$. bounded and simplified to $r(s,a)$

The process generates **trajectories** $h_t = (s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t)$, with $s_{t+1} \sim p(\cdot|s_t, a_t)$

# Example: Tabular MDP




- $\mathcal{S} = \{s_0, s_1, s_2\}$
- $\mathcal{A} = \{a_0, a_1\}$ $(\mathcal{A}_{s_0} = \{a_0\}, \mathcal{A}_{s_1} = \{a_0\}, \mathcal{A}_{s_1} = \{a_0, a_1\})$
- Mean reward in $s_1$

$$r(s_1, a_0) = 2/3 \qquad r(s_1, a_1) = 0$$

- Transition probabilities in $s_0$ by taking action $a_0$

$$p(s_1|s_0, a_0) = 0.1 \qquad p(s_2|s_0, a_0) = 0.9$$

# Markov Decision Process: the Assumptions

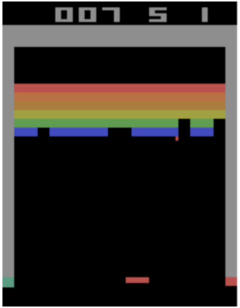
*Markov assumption*: the current state $s$ and action $a$ are a sufficient statistics for the next state $s'$

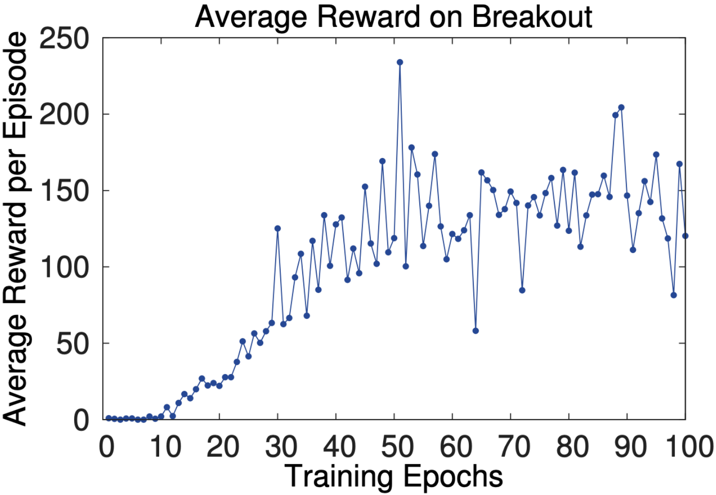
$$p(s'|s,a) = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$$

*Possible relaxations*

- Possible to extend to continuous state-action space
- Define a new state $x_t = (s_t, s_{t-1}, s_{t-2}, \ldots)$ (i.e., $k$-order MDP)
- Move to partially observable MDP (PO-MDP)
- Move to predictive state representation (PSR) model

# ATARI Breakout




* figure from [5]

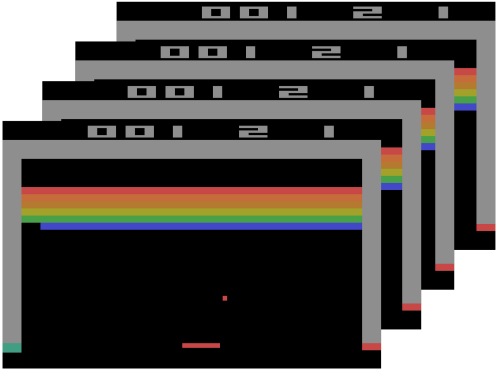
# ATARI Breakout

$\mathbb{P}\Big[ s_{t+1} =$  $\Big| \, s_t =$  $, \text{no-move} \Big]$

Non-Markov dynamics

* figure from [5]

# ATARI Breakout

$$s_t = \left\{ \vphantom{\int} \right.$$  $$\left. \vphantom{\int} \right\}$$

4 consecutive frames = single observation

# Markov Decision Process: the Assumptions

*Time assumption*: time is discrete

$$t \to t + 1$$

*Possible relaxations*

- Identify the proper time granularity
- Most of MDP literature extends to continuous time

# ATARI Breakout



$$a_t = \text{left}$$

$t$

# ATARI Breakout

$a_t = \text{left}$


$t+1$

Too fine-grained resolution

# ATARI Breakout

$$a_t = \text{left}$$

$$t$$

# ATARI Breakout

$a_t = \text{left}$


$t + 1$

Too coarse-grained resolution

# Markov Decision Process: the Assumptions

*Reward assumption*: the reward is uniquely defined by a transition (or part of it)

$$r(x, a, y)$$

*Possible relaxations*

- Distinguish between global goal and reward function
- Move to inverse reinforcement learning (IRL) to induce the reward function from desired behaviors

# Markov Decision Process: the Assumptions

*Stationarity assumption*: the dynamics and reward do not change over time

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a) \qquad r(x,a,y)$$

*Possible relaxations*

- Identify and remove the non-stationary components (e.g., cyclo-stationary dynamics)
- Identify the time-scale of the changes
- Work on finite horizon problems

# Example: the Retail Store Management Problem

*Description.* At each month $t$, a store contains $s_t$ *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 the supplier. Furthermore we know that

- The *cost* of maintaining an inventory of $s$ is $h(s)$.
- 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 $s$, customers that cannot be served leave.
- The *value of the remaining inventory* at the end of the year is $g(s)$.
- *Constraint*: the store has a maximum capacity $M$.
- *Goal:* maximize some measure of profit

# Example: the Retail Store Management Problem

- *State space*: $s \in S = \{0, 1, \dots, M\}$.

# Example: the Retail Store Management Problem

- *State space*: $s \in S = \{0, 1, \ldots, M\}$.
- *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 state $s$, $a \in A(s) = \{0, 1, \ldots, M - s\}$.

# Example: the Retail Store Management Problem

- *State space*: $s \in S = \{0, 1, \ldots, M\}$.
- *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 state $s$, $a \in A(s) = \{0, 1, \ldots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.
  **Problem**: the dynamics should be Markov and stationary!

# Example: the Retail Store Management Problem

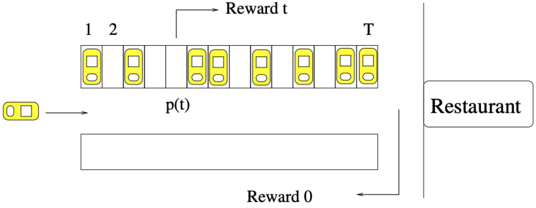
- *State space*: $s \in S = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state $s$, $a \in A(s) = \{0, 1, \dots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.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

# Example: the Retail Store Management Problem

- *State space*: $s \in S = \{0, 1, \ldots, M\}$.
- *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 state $s$, $a \in A(s) = \{0, 1, \ldots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.
  **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*: $r_t = -C(a_t) - h(s_t + a_t) + f([s_t + a_t - s_{t+1}]^+)$.

# Exercise: the Parking Problem

A driver wants to park his car as close as possible to the restaurant.




- The driver cannot see whether a place is available unless he is in front of it.
- There are $P$ places.
- At each place $i$ the driver can either move to the next place or park (if the place is available).
- The closer to the restaurant the parking, the higher the satisfaction.
- If the driver doesn't park anywhere, then he/she leaves the restaurant and has to find another one.

# Policy



## Definition (Policy)

A *decision rule* $d$ can be

- *Deterministic*: $d : S \to A$,
- *Stochastic*: $d : S \to \Delta(A)$ ,
- *History-dependent*: $d : H_t \to A$,
- *Markov*: $d : S \to \Delta(A)$,

Probability distribution over actions

Space of histories
$h_t = (s_1, a_1, s_2, \ldots, s_t)$

# Policy



**Definition (Policy)**

A *decision rule* $d$ can be

- *Deterministic*: $d : S \to A$,
- *Stochastic*: $d : S \to \Delta(A)$ ,
- *History-dependent*: $d : H_t \to A$,
- *Markov*: $d : S \to \Delta(A)$,

Probability distribution over actions

Space of histories $h_t = (s_1, a_1, s_2, \dots, s_t)$

A *policy* (strategy, plan) can be

- *Non-stationary*: $\pi = (d_0, d_1, d_2, \dots)$,
- *Stationary*: $\pi = (d, d, d, \dots)$.

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

$$a_t \sim d_t(s_t)$$

# Example: the Retail Store Management Problem

- Stationary policy composed of deterministic Markov decision rules

$$\pi(s) = \begin{cases} M - s & \text{if } s < M/4 \\ 0 & \text{otherwise} \end{cases}$$

- Stationary policy composed of stochastic history-dependent decision rules

$$\pi(s_t) = \begin{cases} \mathcal{U}(M - s, M - s + 10) & \text{if } s_t < s_{t-1}/2 \\ 0 & \text{otherwise} \end{cases}$$

- Non-stationary policy composed of deterministic Markov decision rules

$$d_t(s) = \begin{cases} M - s & \text{if } t < 6 \\ \lfloor (M - s)/5 \rfloor & \text{otherwise} \end{cases}$$

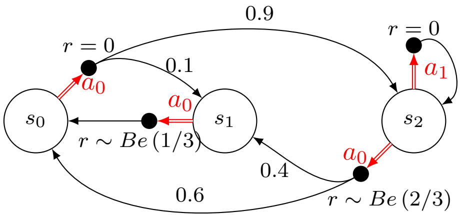
# Markov Chain of a Policy

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

$$P^\pi(s'|s) = \mathbb{P}(s_{t+1} = s'|s_t = s, \pi) = \sum_{a\in\mathcal{A}} \pi(s,a)p(s'|s,a)$$

**Example:** only 2 deterministic stationary policies

$\pi_0 = \{a_0, a_0, a_0\}$
$\pi_1 = \{a_0, a_0, a_1\}$



# Markov Chain of a Policy

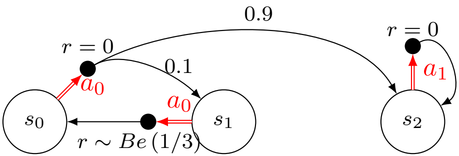


Under a stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, the random process $(s_t)_{t\in\mathbb{N}}$ is a *Markov Chain*, with transition probability

$$P^\pi(s'|s) = \mathbb{P}(s_{t+1} = s'|s_t = s, \pi) = \sum_{a\in\mathcal{A}} \pi(s,a)p(s'|s,a)$$

**Example:** only 2 deterministic stationary policies

$\pi_0 = \{a_0, a_0, a_0\}$
$\pi_1 = \{a_0, a_0, a_1\}$



# Markov Chain of a Policy

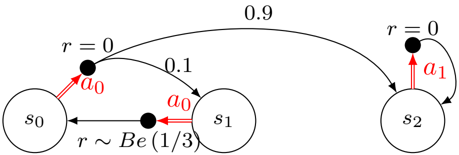


Under a stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, the random process $(s_t)_{t \in \mathbb{N}}$ is a *Markov Chain*, with transition probability

$$P^\pi(s'|s) = \mathbb{P}(s_{t+1} = s'|s_t = s, \pi) = \sum_{a \in \mathcal{A}} \pi(s,a) p(s'|s,a)$$

**Example:** only 2 deterministic stationary policies

$\pi_0 = \{a_0, a_0, a_0\}$
$\pi_1 = \{a_0, a_0, a_1\}$




A MDP is sometimes referred as controlled Markov chain

What is the "utility" of a policy?

i.e., how good is a policy

# State Value Function

Given a policy $\pi = (d_1, d_2, \dots,)$ (deterministic to simplify notation)

- *Finite time horizon $T$*: deadline at time $T$, the agent focuses on the sum of the rewards up to $T$.

$$V^\pi(t,s) = \mathbb{E}\left[ \sum_{\tau=t}^{T-1} r(s_\tau, d_\tau(h_\tau)) + R(s_T) \middle| \, s_t = s; \pi = (d_1, \dots, d_T) \right],$$

where $R$ is a value function for the final state.

# State Value Function



Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Finite time horizon $T$*: deadline at time $T$, the agent focuses on the sum of the rewards up to $T$.

$$V^\pi(t,s) = \mathbb{E}\left[ \sum_{\tau=t}^{T-1} r(s_\tau, d_\tau(h_\tau)) + R(s_T) \middle| \, s_t = s; \pi = (d_1, \dots, d_T) \right],$$

  where $R$ is a value function for the final state.

- *Used when:* there is an intrinsic deadline to meet.

# State Value Function



Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, d_t(h_t)) \,|\, s_0 = s; \pi\right],$$

with discount factor $0 \le \gamma < 1$:

- *small* = short-term rewards, *big* = long-term rewards
- for any $\gamma \in [0, 1)$ the series always converge (for bounded rewards)

# State Value Function



Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, d_t(h_t)) \,|\, s_0 = s; \pi\right],$$

  with discount factor $0 \leq \gamma < 1$:
  - *small* = short-term rewards, *big* = long-term rewards
  - for any $\gamma \in [0, 1)$ the series always converge (for bounded rewards)
- *Used when:* there is uncertainty about the deadline and/or an intrinsic definition of discount.

# State Value Function

Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Stochastic shortest path*: the problem never terminates but the agent will eventually reach a *termination state*.

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{T_\pi} r(s_t, d_t(h_t)) | s_0 = s; \pi\right],$$

where $T_\pi$ is the first (*random*) time when the *termination state* is achieved.

# State Value Function

Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Stochastic shortest path*: the problem never terminates but the agent will eventually reach a *termination state*.

$$V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{T_\pi} r(s_t, d_t(h_t)) | s_0 = s; \pi\Big],$$

  where $T_\pi$ is the first (*random*) time when the *termination state* is achieved.
- *Used when:* there is a specific goal condition.

# State Value Function

Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Infinite time horizon with average reward*: the problem never terminates but the agent only focuses on the (expected) *average of the rewards*.

$$\rho^\pi(s) = \lim_{T \to \infty} \mathbb{E}\left[ \frac{1}{T} \sum_{t=0}^{T-1} r(s_t, d_t(h_t)) \,\middle|\, s_0 = s; \pi \right].$$

# State Value Function

Given a policy $\pi = (d_1, d_2, \dots, )$ (deterministic to simplify notation)

- *Infinite time horizon with average reward*: the problem never terminates but the agent only focuses on the (expected) *average of the rewards*.

$$\rho^\pi(s) = \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} r(s_t, d_t(h_t)) \,|\, s_0 = s; \pi\right].$$

- *Used when:* the system should be constantly controlled over time.

# State Value Function

*Technical note*: the expectations refer to all possible stochastic trajectories.
A (possibly non-stationary stochastic) policy $\pi$ applied from state $s_0$ returns

$$(s_0, r_0, s_1, r_1, s_2, r_2, \ldots)$$

where $r_t = r(s_t, d_t(h_t))$ and $s_t \sim p(\cdot|s_{t-1}, a_{t-1} = d_{t-1}(h_{t-1}))$ are *random* realizations.

The value function (discounted infinite horizon) is

$$V^\pi(s) = \mathbb{E}_{(s_1,s_2,\ldots)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, d_t(h_t)) \,|\, s_0 = s; \pi\right]$$

# Example: the Retail Store Management Problem

Simulation

# Optimization Problem



**Definition (Optimal policy and optimal value function)**

The solution to an MDP is an *optimal policy* $\pi^\star$ satisfying

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

in all the states $x \in X$, where $\Pi$ is some policy set of interest.

# Optimization Problem



**Definition (Optimal policy and optimal value function)**

The solution to an MDP is an *optimal policy* $\pi^\star$ satisfying

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

in all the states $x \in X$, where $\Pi$ is some policy set of interest.

The corresponding value function is the *optimal value function*

$$V^\star = V^{\pi^\star}$$

# Optimization Problem



**Preview of next chapter**

1. $\pi^\star \in \arg\max(\cdot)$ and not $\pi^\star = \arg\max(\cdot)$ because an MDP may admit **more than one** optimal policy
2. $\pi^\star$ achieves the largest possible value function in **every** state
3. there always exists an optimal **deterministic** policy
4. except for finite-horizon problems, there always exists an optimal **stationary** policy
5. there exist **efficient** algorithms to compute value function and optimal policies

# Limitations: Average Case



1. All the previous value functions define an objective **in expectation**
2. Other **utility functions** may be used
3. Risk measures could be integrated but they may induce “weird” problems and make the solution more difficult

# Summary

1. Definition of the Markov Decision Process and its assumptions
2. Definition of a policy
3. Definition of value functions

# Bibliography


[1] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, N.J., 1957.

[2] D.P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[3] W. Fleming and R. Rishel. *Deterministic and stochastic optimal control*. Applications of Mathematics, 1, Springer-Verlag, Berlin New York, 1975.

[4] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[6] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, Etats-Unis, 1994.

# Thank you!

facebook
Artificial Intelligence Research