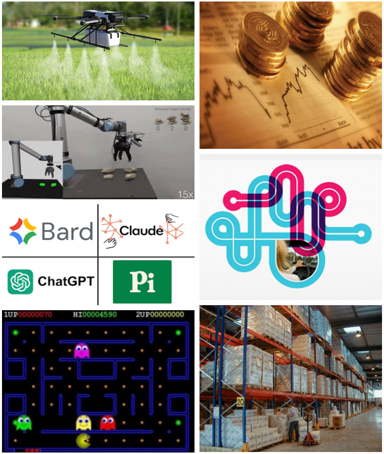
Markov Decision Processes: General Model

Mohammad Sadegh Talebi
m.shahi@di.ku.dk
Department of Computer Science

Sequential Decision Making

Many tasks in real life are **online sequential decision-making** tasks that fall in the framework of **reinforcement learning**:



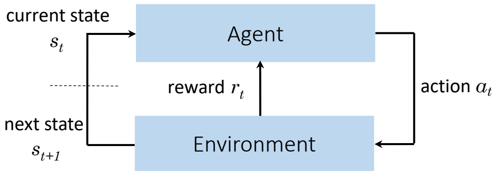
- Selling or buying an asset
- Inventory management
- Portfolio optimization
- Robotics
- Playing computer games
- Routing in networks
- Precision Agriculture and Farming
- LLMs

Sequential Decision Making: General Setting

Almost all RL systems try to solve underlying **decision process**.

Minimal ingredients of a decision process:

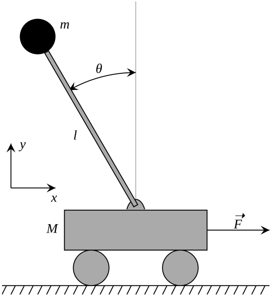
- A notion of **state** capturing different situations
- **Actions** capturing options available at any situation
- A **reward signal** indicating the quality of the action taken



Goal: To maximize an objective function, often defined in terms of rewards $r_1, r_2, \ldots$.

- E.g., maximize $\sum_{t=1}^{N} r_t$ or $\sum_{t=1}^{N} \log(1 + r_t)$

Example: Balancing Cart-pole



Task: Make the pole upright for as long as possible.

- Notions of state:
  - position x
  - position and angle (x, θ)
  - position, angle, velocity, angular velocity $(x, \dot{x}, \theta, \dot{\theta})$
- Action: Force F
- Reward: 1 is $\theta < \theta_{\text{th}}$, else 0. (E.g., th $= 10\,\text{deg}$)

Sequential Decision Making: General Setting

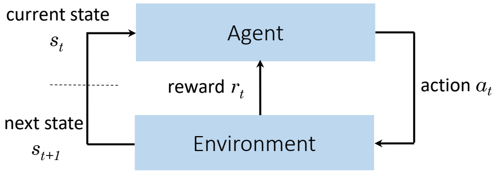
We consider discrete time systems, where time is divided into slots of equal length.

At each time $t = 1, 2, \ldots, N$, an agent interacts with an unknown environment

- observes state s_t,
- chooses an action a_t from a given action set, using a control policy

$$a_t = \text{policy}(s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}),$$

- receives (random) reward r_t.



- **Goal:** To maximize a function of rewards $r_1, r_2, \ldots, r_N$
- Observations and rewards are generated by an **uncertain** and (potentially) **unknown** environment.

Markov Property

Different decision processes differ mainly on how s_{t+1} and r_t are generated.

Under the Markov property, s_{t+1} and r_t only depend on s_t and a_t.

$$\mathbb{P}\Big(s_{t+1} = s' \Big| s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t, a_t\Big) = \mathbb{P}\Big(s_{t+1} = s' \Big| s_t, a_t\Big)$$
$$\mathbb{P}\Big(r_t \Big| s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t, a_t\Big) = \mathbb{P}\Big(r_t \Big| s_t, a_t\Big)$$

Namely, conditioned on (s_t, a_t), the process is independent from the past.

This property defines Markov Decision Processes (MDPs).

Markov Decision Processes

Markov Decision Process

A finite Markov Decision Process (MDP) is a tuple $M = (\mathcal{S}, \mathcal{A}, P, R)$:

- State-space $\mathcal{S}$ (with size S)
- Action-space $\mathcal{A}$ (with size A)
- Transition function P: Selecting $a \in \mathcal{A}$ in $s \in \mathcal{S}$ leads to a transition to s' with probability $P(s'|s,a)$. $P(\cdot|s,a)$ is a probability distribution over $\mathcal{S}$, i.e.,

$$\sum_{s' \in \mathcal{S}} P(s'|s,a) = 1$$

- Reward function R: Selecting $a \in \mathcal{A}$ in $s \in \mathcal{S}$ yields a reward $r \sim R(s,a)$.

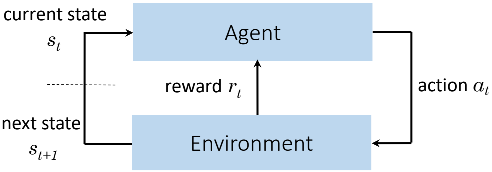
- The action-space may generally be state-dependent; we use $\mathcal{A}_s$ to denote the set of actions available in state s.
- In general, $\mathcal{S}$ or $\mathcal{A}$ could be finite, countably infinite, or continuous.

Interaction with MDP

An **agent** interacts with the MDP for N rounds.

At each time step t:

- The agent observes the current state s_t and takes an action $a_t \in \mathcal{A}$
- The environment (MDP) decides a reward $r_t := r(s_t, a_t) \sim R(s_t, a_t)$ and a next state $s_{t+1} \sim P(\cdot|s_t, a_t)$
- The agent receives r_t (any time in step t before start of $t + 1$)



This interaction produces a trajectory (or history)

$$h_t = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$$

Markov Property

MDPs adhere to the Markov property.

- At each time t, s_{t+1} and r_t only depend on s_t and a_t.
- More precisely,

$$\mathbb{P}\Big(s_{t+1} = s' \Big| s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t, a_t\Big) = \underbrace{\mathbb{P}\Big(s_{t+1} = s' \Big| s_t, a_t\Big)}_{=P(s'|s_t, a_t)}$$

$$R\big(s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t, a_t\big) = R(s_t, a_t)$$

Classification of MDPs based on Horizon N

- **Finite-Horizon MDPs**: $N < \infty$, and the goal is to solve

$$\max_{\text{all strategies}} \mathbb{E}\Big[\sum_{t=1}^{N-1} r(s_t, a_t) + r(s_N)\Big]$$

- **Infinite-Horizon Discounted MDPs:** $N = \infty$, and given discount factor $\gamma \in (0,1)$, the goal is to solve

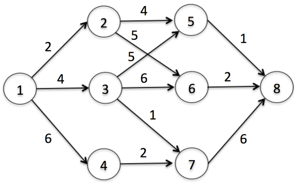
$$\max_{\text{all strategies}} \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)\Big]$$

- **Infinite-Horizon Undiscounted MDPs (Average-Reward MDPs):** $N = \infty$, and the goal is to solve

$$\max_{\text{all strategies}} \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\Big[\sum_{t=1}^{N} r(s_t, a_t)\Big]$$

MDP Examples

Example: Routing



Task: Find the maximum-weight route between node 1) and destination (node 8).

Modeling as finite-horizon MDP:

- States: Nodes in the graph $\mathcal{S} = \{1, 2, \ldots, 8\}$
- Actions: Outgoing edges at each state; e.g., $\mathcal{A}_2 = \{\text{go to 4}, \text{go to 5}\}$
- Deterministic transitions
- Rewards: Edge weights
- Time horizon N: any number greater than the maximum path length $(N \geq 4)$

Example: Product Management

Suppose we receive an order for a given product with probability α. We can either process all the unfilled orders or process no order.

- The cost per unfilled order per period is $c > 0$, and the setup cost to process unfilled order is $K > 0$.
- Assume that the total number of orders that can remain unfilled is n.

Task: Find an order processing strategy that has minimal expected cost.

Example: Product Management

Modeling as a discounted MDP:

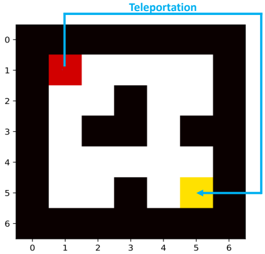
- **State Space:** Define the state as the number of unfilled orders at the beginning of each period $\implies \mathcal{S} = \{0, 1, \dots, n\}$.
- **Action Space:** For $s \neq 0, n$, we have $\mathcal{A}_s = \{J, \overline{J}\}$, where $J =$processing unfilled orders and $\overline{J}$ = processing no order $\implies \mathcal{A}_0 = \{\overline{J}\}$ and $\mathcal{A}_n = \{J\}$.
- **Reward Function:**

$$R(i, J) = -K, \quad R(i, \overline{J}) = -ci, \quad i = 1, \dots, n-1,$$
$$R(0, \overline{J}) = 0, \quad R(n, J) = -K.$$

- **Transition Function:**

$$P(0|i, J) = 1 - \alpha, \quad P(1|i, J) = \alpha, \quad i = 1, 2, \dots, n-1,$$
$$P(i|i, \overline{J}) = 1 - \alpha, \quad P(i+1|i, \overline{J}) = \alpha, \quad i = 1, 2, \dots, n-1,$$
$$P(0|n, J) = 1 - \alpha, \quad P(1|n, J) = \alpha,$$
$$P(0|0, \overline{J}) = 1 - \alpha, \quad P(1|0, \overline{J}) = \alpha.$$

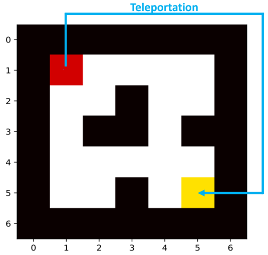
Example: Grid-world

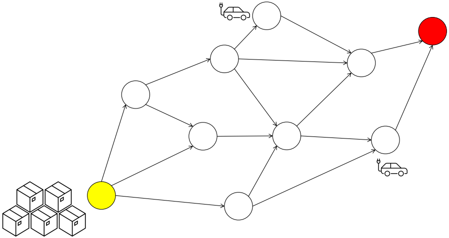


Task: Find the shortest path from yellow to red.

- A grid-world with $S = 20$ states, and 4 actions $(\rightarrow, \uparrow, \downarrow, \leftarrow)$.
- E.g., $a = \uparrow$ yields: moving $\uparrow$ (w.p. 0.7), no move (w.p. 0.1), or moving $\rightarrow$ or $\leftarrow$ (each w.p. 0.1)
- Reward is 1 in red, else 0.
- Once in red:
  - the agent may stay there forever (**one-shot** task), or
  - the agent may be teleported to yellow (**continual** task)

Example: Grid-world

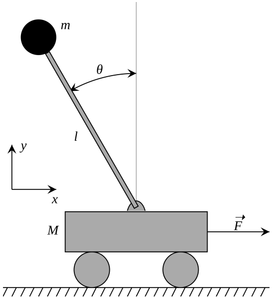




Task: Find the shortest path from to .

- A grid-world with $S = 20$ states, and 4 actions $(\rightarrow, \uparrow, \downarrow, \leftarrow)$.
- E.g., $a = \uparrow$ yields: moving $\uparrow$ (w.p. 0.7), no move (w.p. 0.1), or moving $\rightarrow$ or $\leftarrow$ (each w.p. 0.1)
- Reward is 1 in , else 0.
- Once in :
  - the agent may stay there forever (**one-shot** task), or
  - the agent may be teleported to (**continual** task)

Example: Balancing Cart-pole



Task: Make the pole upright for as long as possible.

- Notion of state? $s = x$ or $s = (x, \theta)$? Neither will yield an MDP definition.
- State: $s = (x, \dot{x}, \theta, \dot{\theta})$
- Action: Force F
- Reward: If $|\theta| >$ th, then 0, else 1. (E.g., th $= 10\,\mathrm{deg}$)

Once $|\theta| >$ th, an episode is terminated, and the pole is put at $\theta = 0$.

Example: Query Optimization

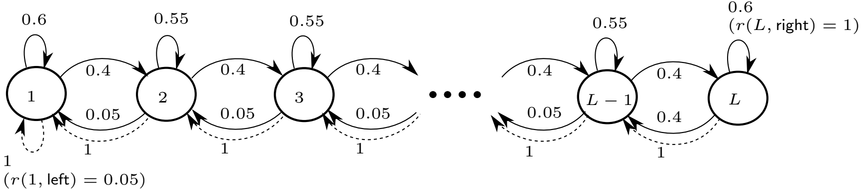
Query Optimization via RL (Marcus et al. (2019); Chen et al. (2023))

Task: Find a query plan minimizing long-term execution time

- State: Partial query plan
- Action: Join order decisions
- Reward: negative of execution cost

Example: RiverSwim

The L-state RiverSwim MDP



Exercise: Determine

- **State Space:**
- **Action Space:**
- **Reward Function:**
- **Transition Function:**

Policy

Policy

When interacting with an MDP, actions are taken according to some **policy**:

Classification of policies:

- deterministic vs. randomized (stochastic)
- stationary vs. history-dependent

	deterministic	randomized
stationary		
history-dependent		

Stationary Policies

A **stationary deterministic** policy π is a mapping $\pi : \mathcal{S} \to \mathcal{A}$.

- Notation: $a = \pi(s)$
- π prescribes an action with certainty at any state s, without dependence on past states or actions.

A **stationary randomized** policy π is a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the set of probability distributions over $\mathcal{A}$.

- Notation: $a \sim \pi(\cdot|s)$ or $\pi(a|s)$ denotes the probability of selecting a in s:

$$\sum_{a \in \mathcal{A}} \pi(a|s) = 1$$

- At any state s, π prescribes a probability distribution over $\mathcal{A}$, but without dependence on past states or actions.

History-dependent Policies

Let $\mathcal{H}$ the set of all possible histories (trajectories).

A **history-dependent deterministic** policy π is a mapping $\pi : \mathcal{H} \to \mathcal{A}$.

- Notation: $a = \pi(h_t)$ at time t
- π prescribes an action with certainty at any state s, but depends on past states or actions.

A **history-dependent randomized** policy π is a mapping $\pi : \mathcal{H} \to \Delta(\mathcal{A})$.

- Notation: $a \sim \pi(\cdot|h_t)$ or $\pi(a|h_t)$ denotes the probability of selecting a given history h_t:

$$\sum_{a \in \mathcal{A}} \pi(a|h_t) = 1, \qquad \forall t.$$

- Given any history h_t, π prescribes a probability distribution over $\mathcal{A}$, arbitrarily depending on past states or actions.

Policy

	deterministic	randomized
stationary	$\pi : \mathcal{S} \to \mathcal{A}$	$\pi : \mathcal{S} \to \Delta(\mathcal{A})$
history-dependent	$\pi : \mathcal{H} \to \mathcal{A}$	$\pi : \mathcal{H} \to \Delta(\mathcal{A})$

- Π^{SD}: The set of stationary deterministic policies
- Π^{SR}: The set of stationary randomized policies
- Π^{HD}: The set of history-dependent deterministic policies
- Π^{HR}: The set of history-dependent randomized policies

(i) $\Pi^{\mathsf{SD}} \subset \Pi^{\mathsf{SR}} \subset \Pi^{\mathsf{HR}}$ (ii) $\Pi^{\mathsf{SD}} \subset \Pi^{\mathsf{HD}} \subset \Pi^{\mathsf{HR}}$

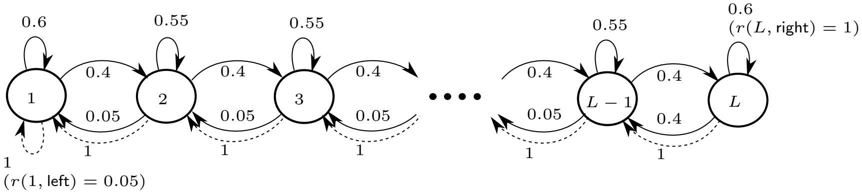
Notation:

- For $\pi \in \Pi^{\mathsf{SR}}$, we write $a \sim \pi(\cdot|s)$. Also, given $f : \mathcal{A} \to \mathbb{R}$,

$$\mathbb{E}_{a\sim\pi(s)}[f(a)] = \sum\nolimits_{a\in\mathcal{A}} f(a)\pi(a|s)$$

Policy: Examples

The L-state RiverSwim MDP



Examples:

- π_1 : always go right. ($\pi_1 \in \Pi^{\mathsf{SD}}$)
- π_2 : go right w.p. 0.7 and left w.p. 0.3. ($\pi_2 \in \Pi^{\mathsf{SR}}$)
- π_3 : go right if $s_t \neq s_{t-1}$, otherwise go left . ($\pi_3 \in \Pi^{\mathsf{HD}}$)

Induced Markov Chains

- Every $\pi \in \Pi^{\mathsf{SR}}$ induces a Markov chain on M, with transition probability matrix P^π given by:

$$P^\pi_{s,s'} = \sum_{a \in \mathcal{A}} P(s'|s,a)\pi(a|s)\,, \quad s, s' \in \mathcal{S}.$$

- Every $\pi \in \Pi^{\mathsf{SR}}$ induces a reward vector $r^\pi \in \mathbb{R}^S$ on M defined by:

$$r^\pi(s) = \sum_{a \in \mathcal{A}} R(s,a)\pi(a|s)\,, \quad s \in \mathcal{S}.$$

- If $\pi \in \Pi^{\mathsf{SD}}$, then $P^\pi_{s,s'} = P(s'|s,\pi(s))$ and $r^\pi(s) = R(s,\pi(s))$.

Every policy $\pi \in \Pi^{\mathsf{SR}}$ induces a **Markov Reward Process (MRP)** on M, specified by r^π and P^π.

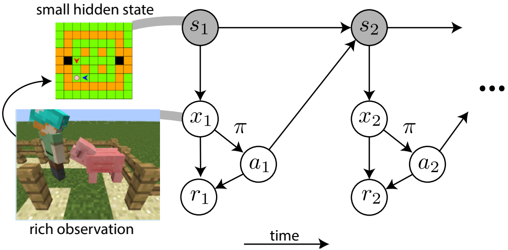
Beyond Full Observability

Partial Observability

- MDPs (and many other decision processes) rest on full observability of the state.
- In some applications, the state is unobservable, but it can be inferred or estimated via some proxy.
- Some related decision processes: Partially Observable MDP (POMDP), Predictive State Representation (PSR), Regular Decision Process.
- RL under partial observability is much more challenging than in MDPs.

RL under partial observability is beyond the scope of OReL.

Example



An image from the Malmo platform built for AI experimentation (Photo from (Dann et al., 2018))

- The task is governed by small but hidden state-space.
- The agent may infer the current state s_t via rich sensory observations encoded as x_t.
- The problem is Markovian w.r.t. s, not x.