CIS 678 Machine Learning

Reinforcement Learning

Plan

- General RL introduction
- Notebook presentation (Mountain Car)

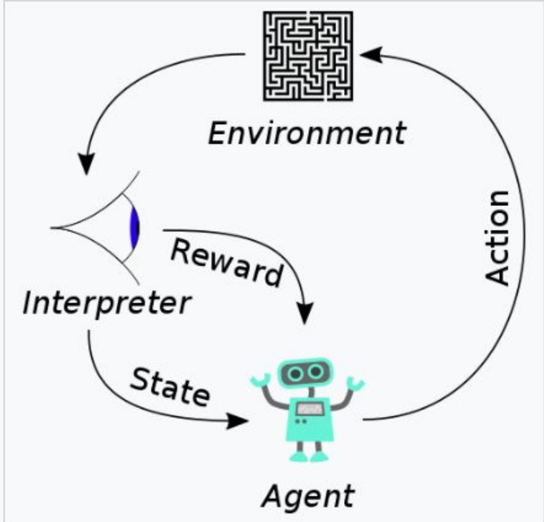
Reinforcement Learning

Problem characteristics:

- Requires sequence of decisions/actions
    - Games such as Chess and Go
    - Solving a maze
    - Robotics
    - Automations
    - ..

Reinforcement Learning

An agent learns by interacting with an environment!



The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent.

Reinforcement Learning

Popular Google DeepMind RL applications

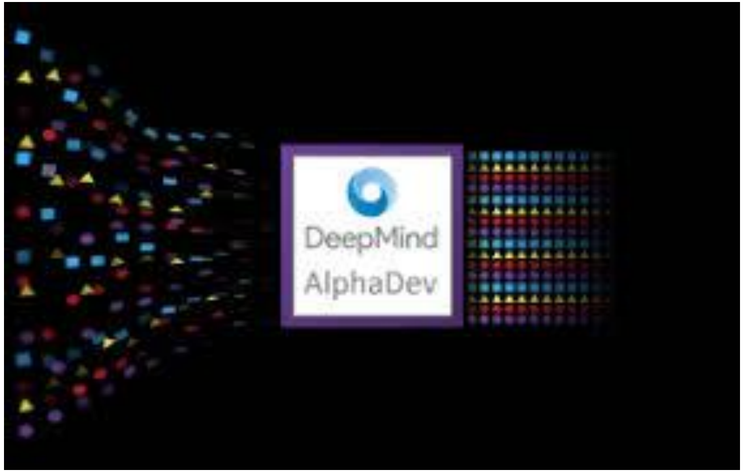
- **AlphaGo**
- Chess
  - Deep Blue (chess) - expert system
  - **AlphaZero (RL)**
  - **Leela Chess Zero**, (LCZero), latest version inspired by **AlphaZero**
- AlphaDev: To discover enhanced computer science algorithms using RL

Reinforcement Learning

Popular Google DeepMind RL applications

- **AlphaGo**
- Chess
  - Deep Blue (chess) - expert system
  - **AlphaZero (RL)**
  - **Leela Chess Zero**, (LCZero), latest version inspired by **AlphaZero**
- AlphaDev: To discover enhanced computer science algorithms using RL

Reinforcement Learning

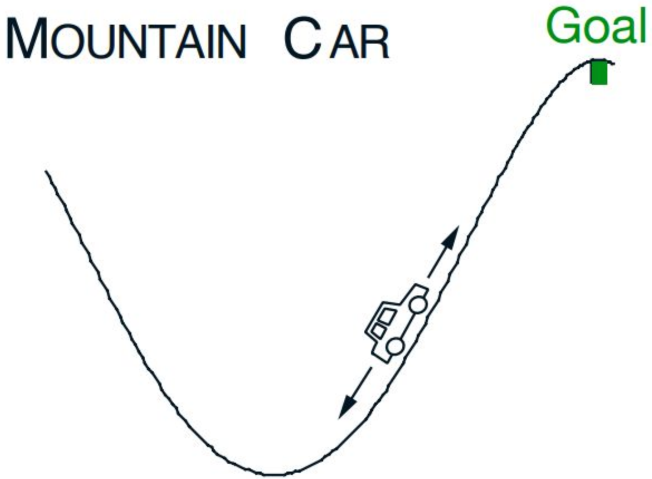


Popular Google DeepMind RL applications

- **AlphaGo**
- Chess
  - Deep Blue (chess) - expert system
  - **AlphaZero (RL)**
  - **Leela Chess Zero**, (LCZero), latest version inspired by **AlphaZero**
- AlphaDev: To discover enhanced computer science algorithms using RL

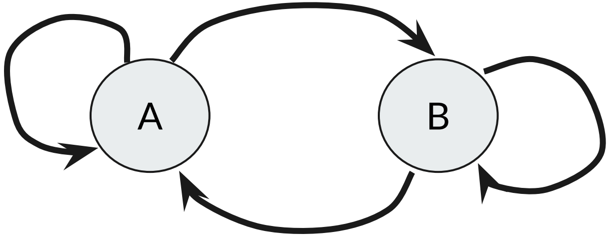
An toy (but difficult) problem

Goal: Reach the top of the mountain (as quickly as possible)



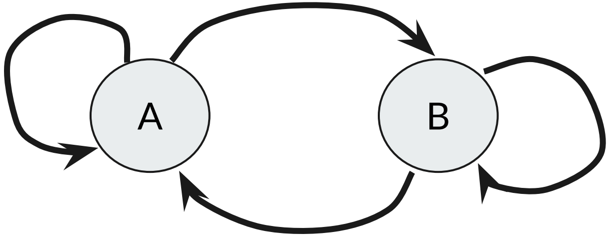
Markov Process -> Markov Decision Process

- Can you think of a two state simple problem?

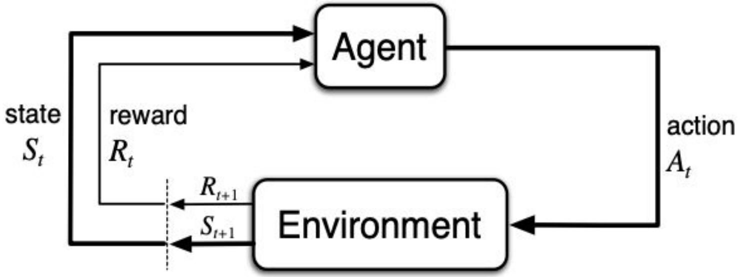


Markov Process -> Markov Decision Process

- Can you think of a two state simple problem?
- A dog likes to always be in his house (B), and only comes to master (at A) when master allows food
  - The master allows food at certain period of the day/night
  - Think it of a time-interval setup, or a time-localization problem (A dog doesn't have a watch; right?)



Some definitions



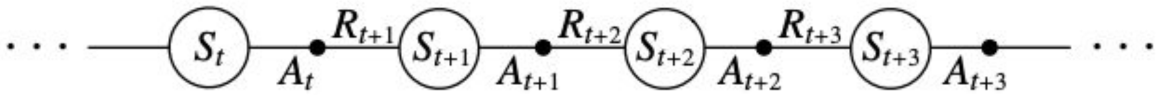
Agent observes state at step t: $S_t \in \mathcal{S}$

produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$

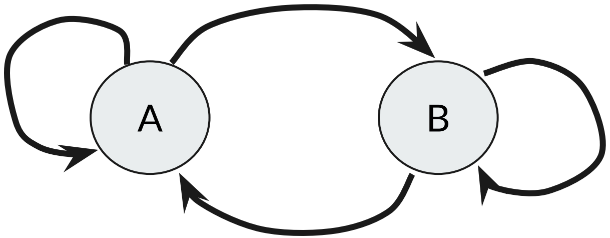
Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$



Adapted from

Markov Decision Process

- If a reinforcement learning task has the Markov Property, it is basically a **Markov Decision Process (MDP)**.
- If state and action sets are finite, it is a **finite MDP**.
- To define a finite MDP, you need to give:
  - **state and action sets**
  - **One-step "dynamics"**



$$\boxed{p(s', r|s, a) = \mathbf{Pr}\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}}$$

The Agents job is to Learn a Policy

Policy at step t = π_t =

a mapping from states to action probabilities

$\pi_t(a \mid s) =$ probability that $A_t = a$ when $S_t = s$

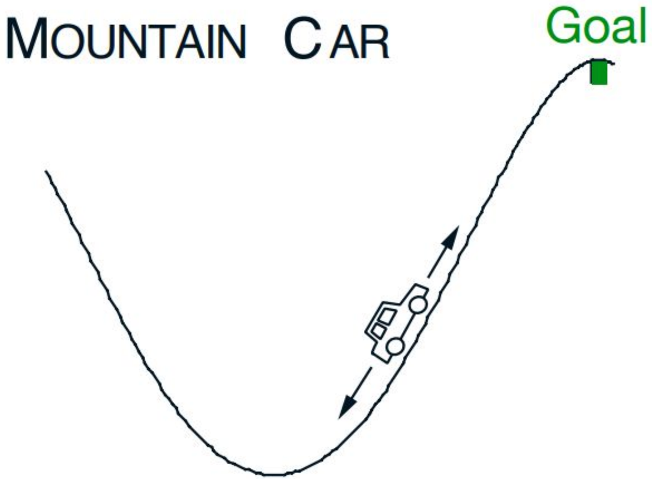
- RL methods specify how the agent changes its policy as a result of experience.
- The agent's goal is to get as much reward as it can over the long run.

Natural phenomenon: Goals, rewards, and returns

Goal: Reach the top of the mountain (as quickly as possible)

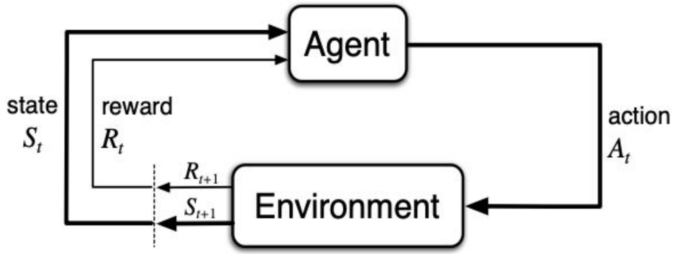
Formulation:

- **Reward** (function) = -1 for each step (state transition)
- (Episode) **return** = - number of steps before reaching the top (goal)
- **Return** (objective) is maximized by minimizing the number of steps (state transition) to reach the top of the mountain



Learning by exploration (episodic Tasks)

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze



In episodic tasks, we almost always use simple total reward:

$$G_t = r_{t+1} + r_{t+2} + .. + r_T$$

where T is a final time step at which a terminal state is reached, ending an episode.

Discounted reward:

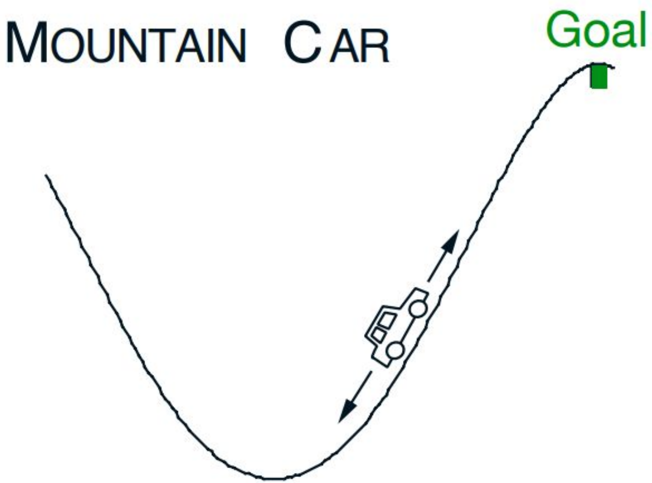
$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

Natural phenomenon: Goals, rewards, and returns

Goal: Reach the top of the mountain (as quickly as possible)

<u>Formulation:</u>

- **Reward** (function) = -1 for each step (state transition)
- (Episode) **return** = - number of steps before reaching the top (goal)
- **Return** (objective) is maximized by minimizing the number of steps (state transition) to reach the top of the mountain



Value function

A **Value function** of a state s is defined as the expected cumulative reward obtained when starting from state s and following policy π.

$$\boxed{V_\pi(s) := E_\pi[G_t|s_t = s] = E_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|s_t = s]}$$

Q function

Value function of a state-action pair or more commonly known as **Q function** of a state-action pair (s,a) is defined as the expected cumulative reward obtained when starting from state s, taking an action a and following policy π there after.

$$\boxed{Q_\pi(s,a) := E_\pi[G_t | s_t = s, a_t = a] = E_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1} | s_t = s, a_t = a]}$$

$$V_\pi(s) := E_\pi[G_t | s_t = s] = E_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1} | s_t = s]$$

The Agents job is to Learn a Policy

Policy at step t = π_t =

a mapping from states to action probabilities

$\pi_t(a \mid s) =$ probability that $A_t = a$ when $S_t = s$

- RL methods specify how the agent changes its policy as a result of experience.
- The agent's goal is to get as much reward as it can over the long run.
- Maximize the **expected reward** and Value/Q function are a tool (part of the algorithm) to achieve this

QA