Reinforcement Learning

PhD course @ “Dottorato di ingegneria industriale e dell’informazione”
Trieste, 2024

Simone Silvetti
PostDoc @ units

Who am I?

Simone Silvetti (simone.silvetti@dia.units.it)

- ➔ Studied mathematics in Rome
- ➔ Phd in Computer Science @ Udine
- ➔ Worked for 11 years in ESTECO
- ➔ Currently PostDoc @ units

application of quantitative formal methods and machine learning techniques to Verification and Model-based Testing of Complex Systems

Who are you?

- ➔ Which is your background?
- ➔ Who knows Machine Learning? Supervised, unsupervised learning?
- ➔ Who knows Reinforcement Learning?

Lessons

- 5/11 14:15 - 16:00 (~15 min break) - Introduction to RL
- 7/11 11:15 - 14:00 (~15 min break)
  - 11:15 - 12:30 - Model free RL
  - 12:30 - 14:00 - Hands on session (plain Python - PyTorch)
- ?? Reinforcement Learning and Temporal Logics

Who am I?

Simone Silvetti (silvetti@esteco.com)

➔ Studied mathematics in Rome

➔ Phd in Computer Science @ Udine

application of quantitative formal methods and machine learning techniques to Verification and Model-based Testing of Complex Systems

➔ Currently working in ESTECO

Numerical Methods Group

multi-objective optimization algorithms, machine learning, object-oriented programming

Who am I?

Simone Silvetti (silvetti@esteco.com)

- ➔ Studied mathematics in Rome
- ➔ Phd in Computer Science @ Udine
- ➔ Currently working in ESTECO

application of quantitative formal methods and machine learning techniques to Verification and Model-based Testing of Complex Systems

Numerical Methods Group — multi-objective optimization algorithms, machine learning, object-oriented programming

Research and Development — process mining, research projects related to technology and domains useful for ESTECO products

Who am I?

Simone Silvetti (silvetti@esteco.com)

- ➔ Studied mathematics in Rome
- ➔ Phd in Computer Science @ Udine
- ➔ Currently working in ESTECO

application of quantitative formal methods and machine learning techniques to Verification and Model-based Testing of Complex Systems

Numerical Methods Group

multi-objective optimization algorithms, machine learning, object-oriented programming

I worked on “Inverse Reinforcement Learning” applied to autonomous driving

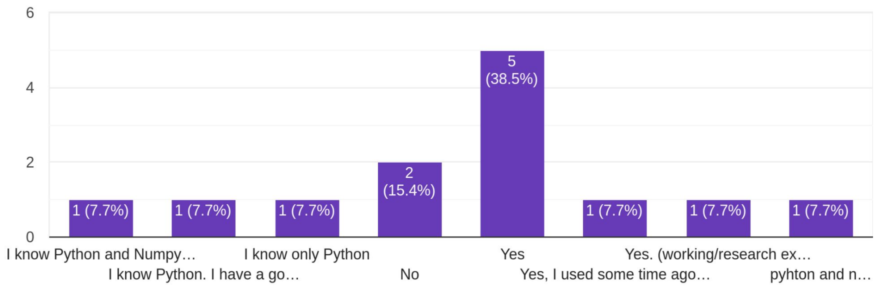
Research and Development

process mining, research projects related to technology and domains useful for ESTECO products

Who are you?

Do you know Python? Numpy, Scipy?

13 responses



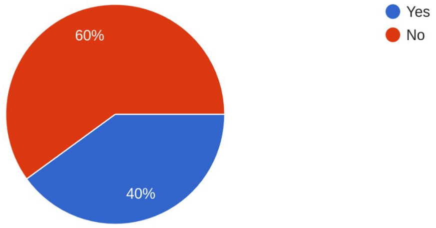
During your studies have you participated in courses of Reinforcement Learning? If yes, which topics have you covered?

13 responses

- No
- Only partially
- I did not partecipate to any course.
- I have never participated in a course about Reinforcement Learning.
- I have never participated at any course of bayesian optimization
- no
- Foundations

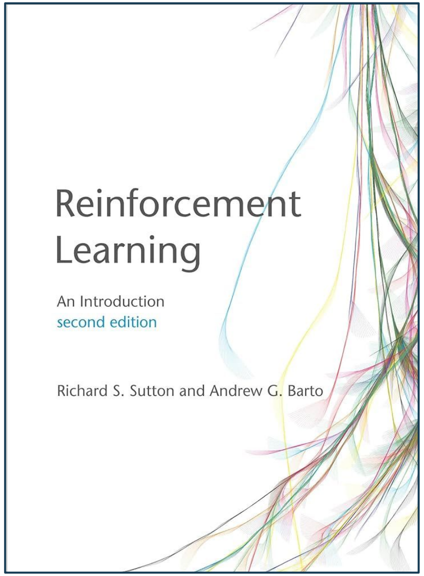
Will you follow the "Learning-based Controllers and the Reality Gap" course?

5 responses



Reference

A book from Sutton et al.

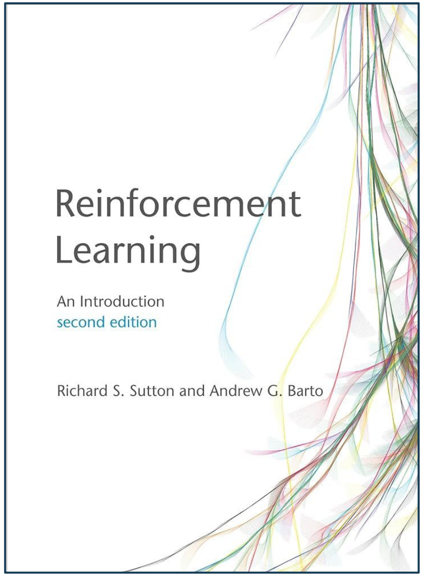


Free available here!

http://incompleteideas.net/book/the-book-2nd.html

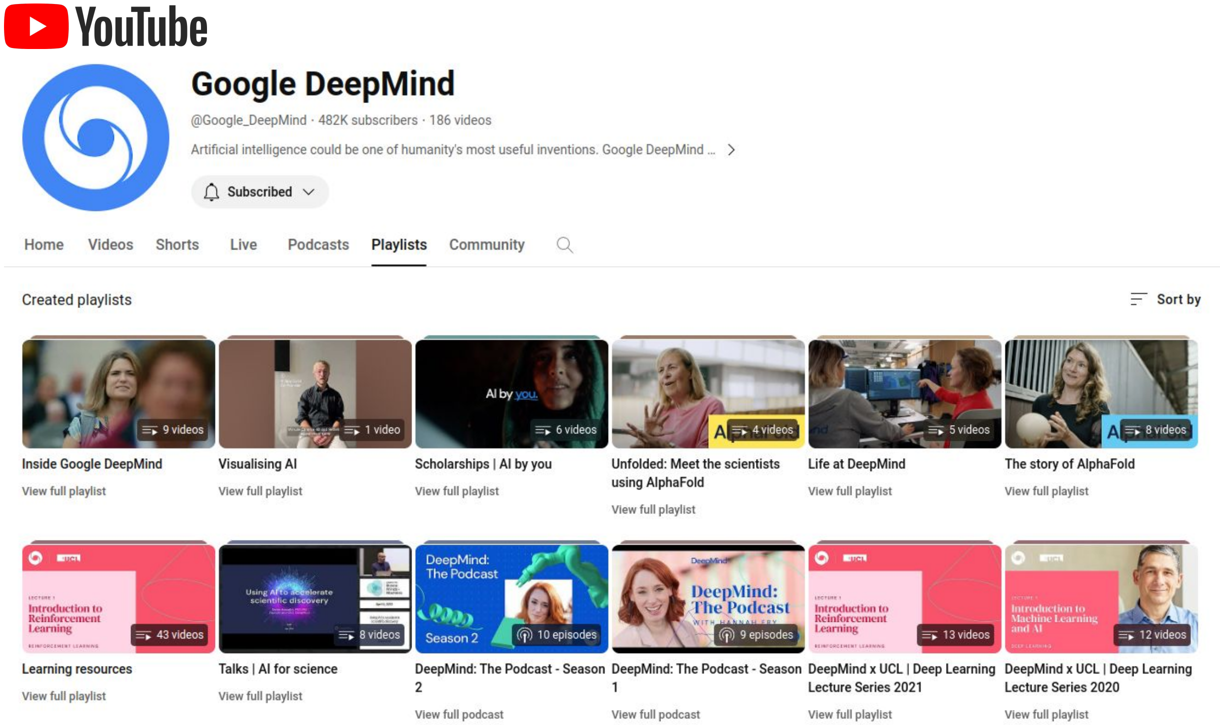
Reference

A book from Sutton et al.



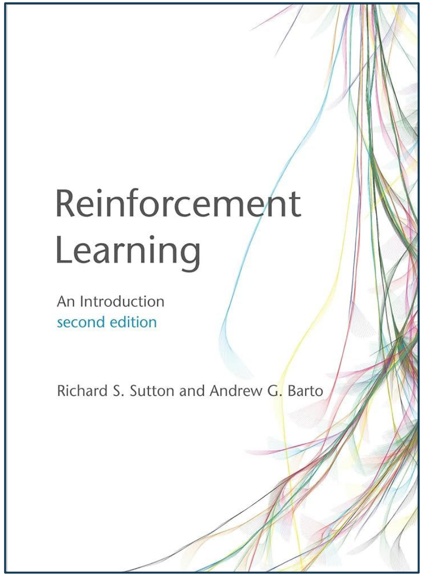
Free available here!

http://incompleteideas.net/book/the-book-2nd.html



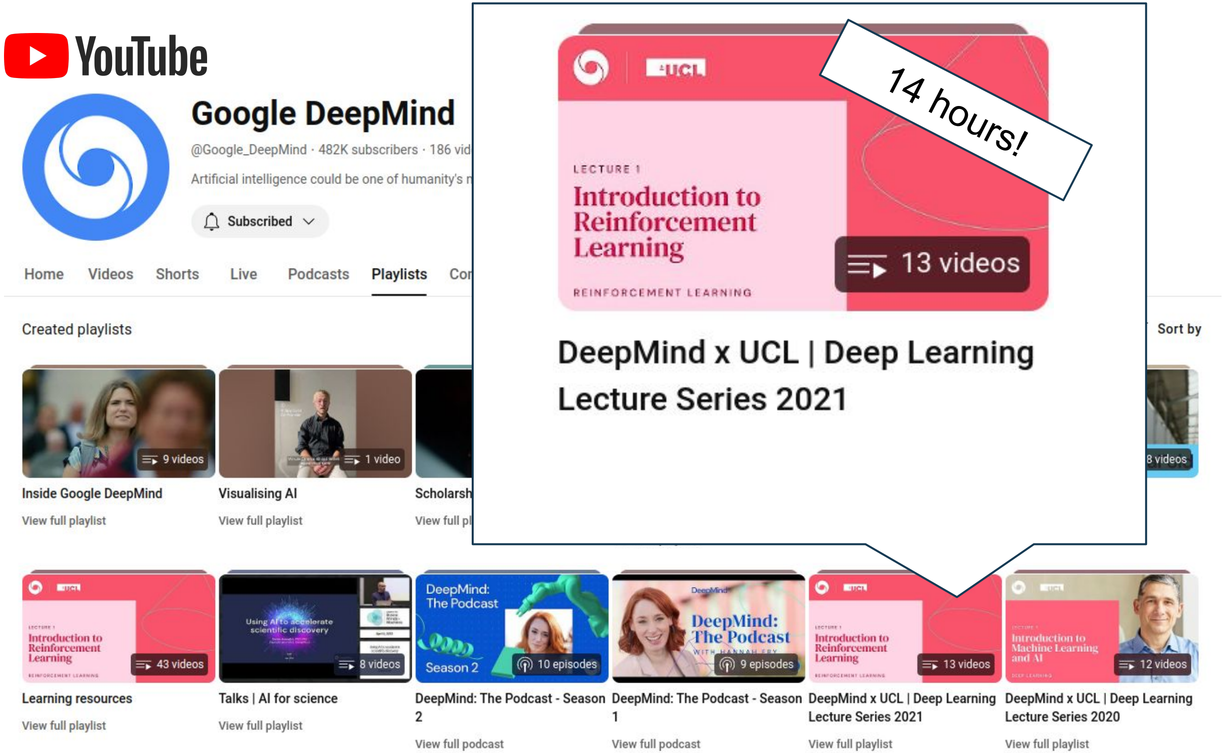
Reference

A book from Sutton et al.



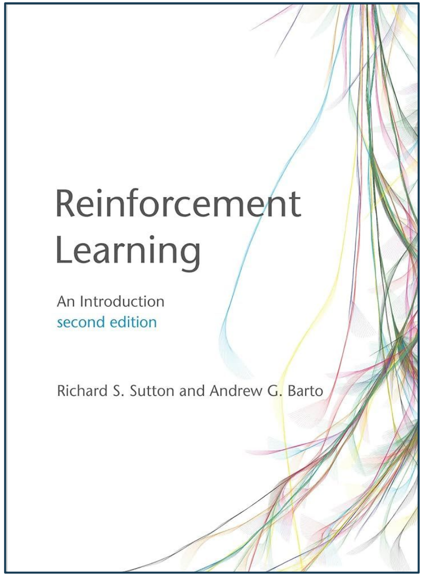
Free available here!

http://incompleteideas.net/book/the-book-2nd.html



Reference

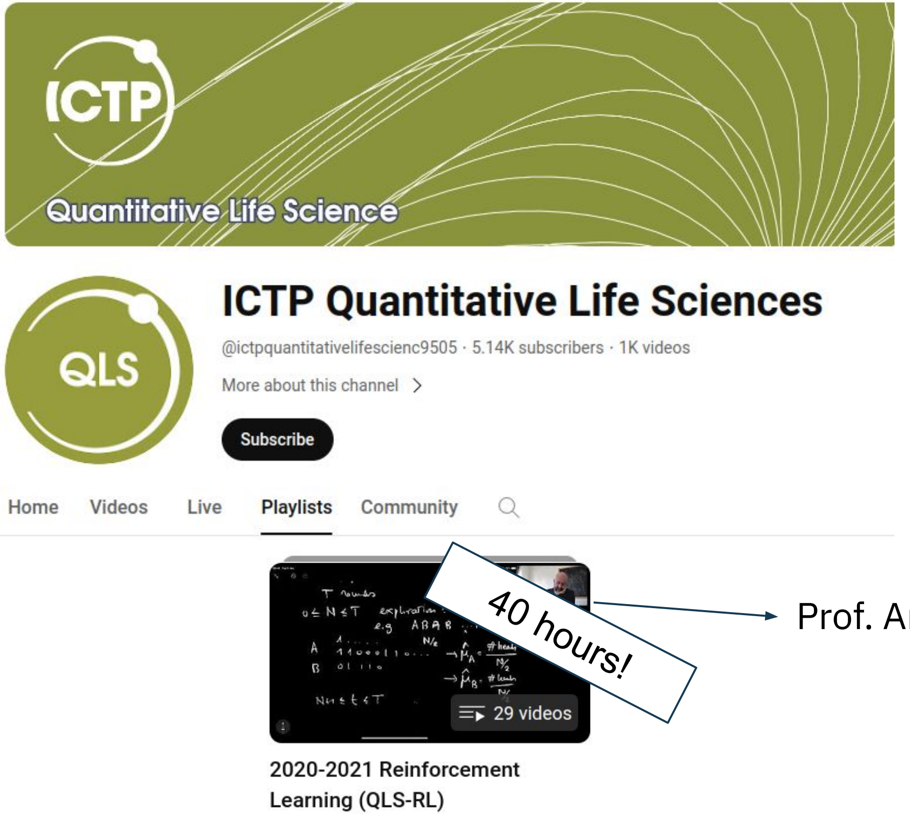
A book from Sutton et al.



Free available here!

http://incompleteideas.net/book/the-book-2nd.html

YouTube



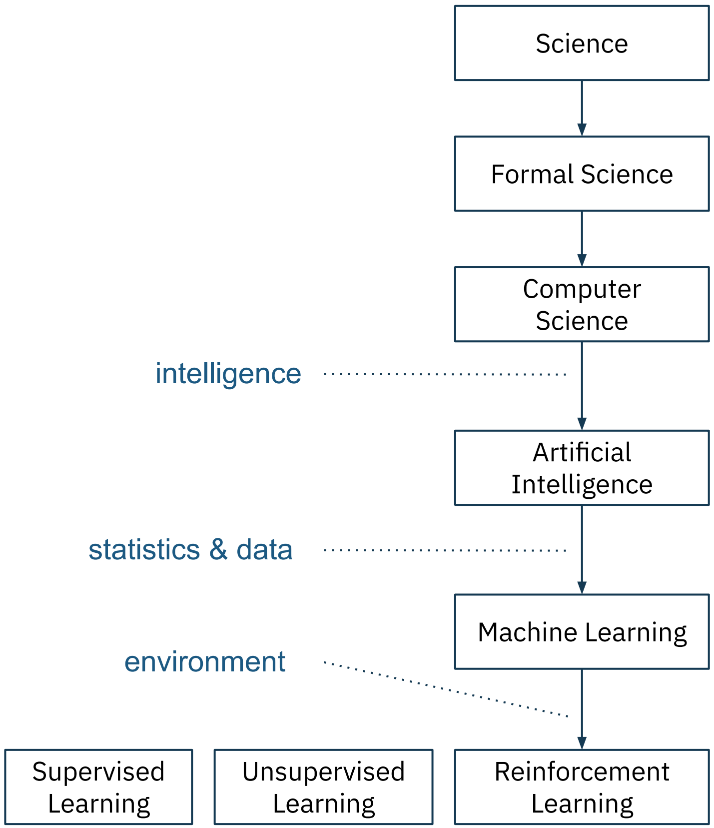
40 hours!

Prof. Antonio Celani

Introduction

What is Reinforcement Learning?

A map



the systematic study of physical and natural world through observation, experimentation, and the testing of theories against the evidence obtained

uses formal systems to generate knowledge

is the study of computation, information and automation

enabling machines to perceive their environment and uses learning and intelligence to take actions that maximize their chances of achieving defined goals

development and study of **statistical algorithms** that can learn from data and **generalize** to unseen data, and thus perform tasks without explicit instructions.

technique that trains software to make decisions to achieve the most optimal results

A definition

Reinforcement Learning	technique that trains software to make decisions to achieve the most optimal results

A definition

Reinforcement Learning

technique that trains ~~software~~ to ~~make decisions~~ to ~~achieve the most optimal results~~

A definition

Reinforcement Learning

technique that trains **agents** to ~~make decisions~~ to ~~achieve the most optimal results~~

A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to ~~achieve the most optimal results~~

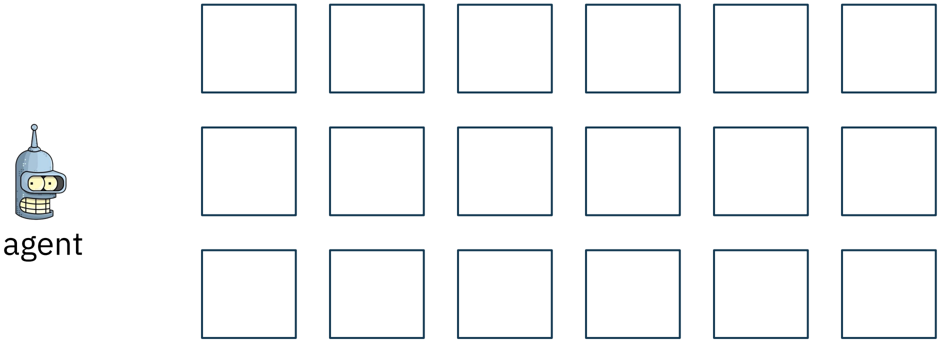
A definition

Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

A definition

Reinforcement Learning

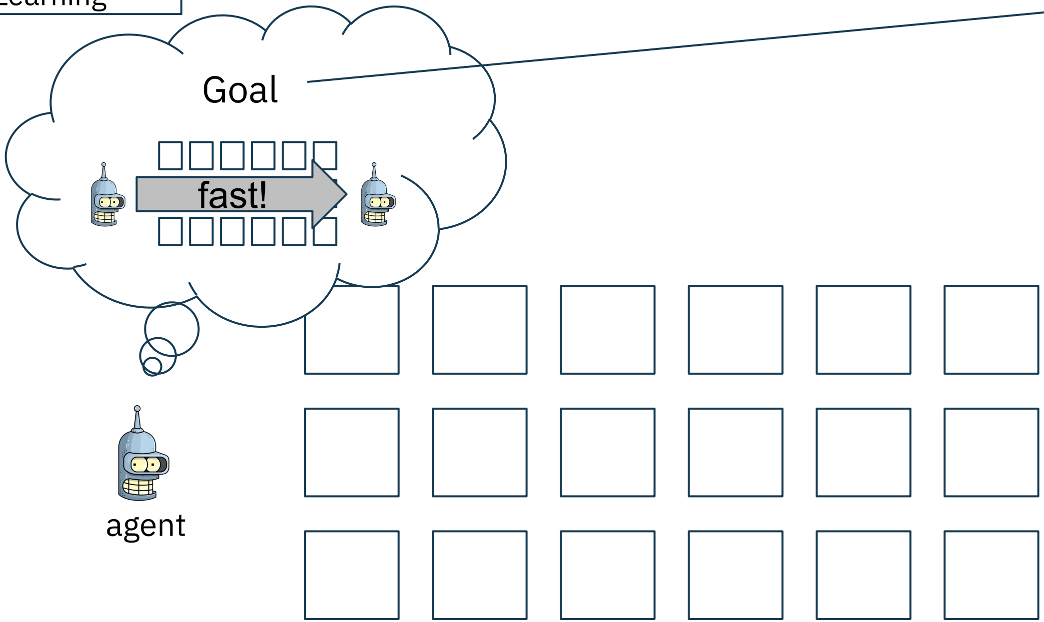
technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



A definition

Reinforcement Learning

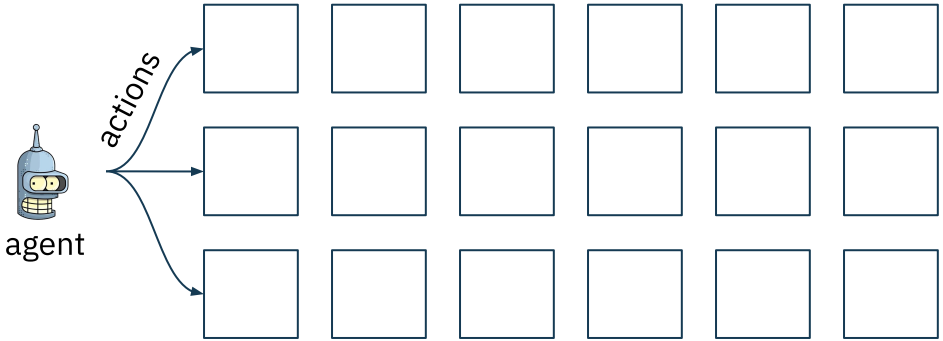
technique that trains **agents** to **map states into actions** to **maximize a <u>cumulative reward</u>**



A definition

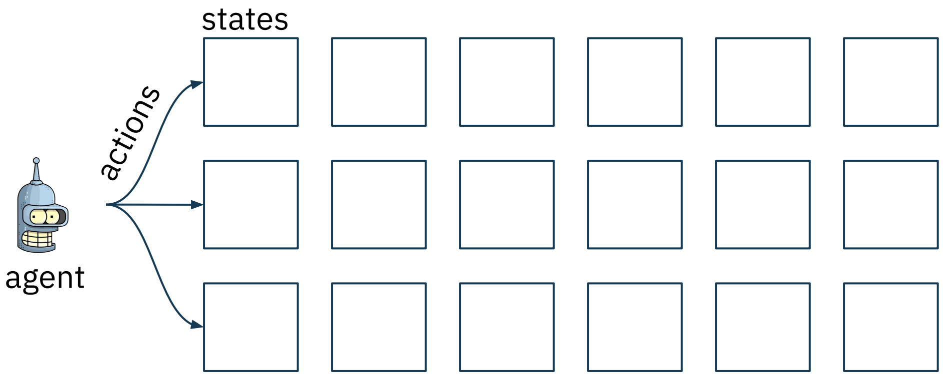
Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



A definition

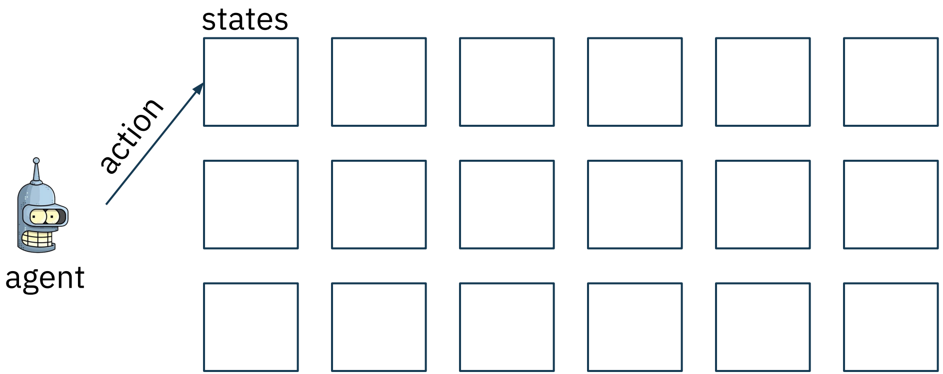
Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



A definition

Reinforcement Learning

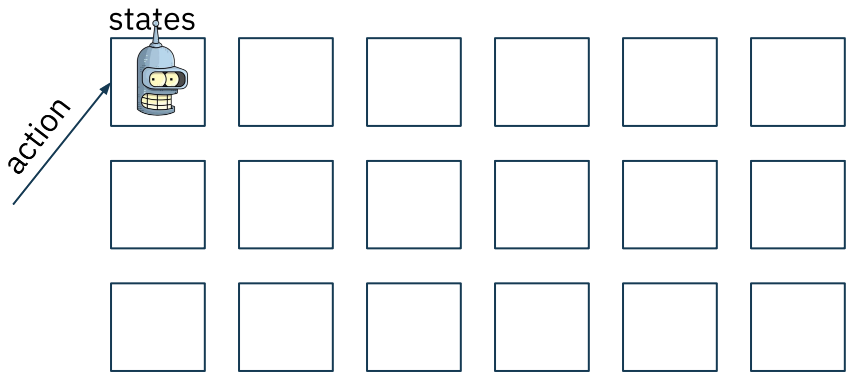
technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



A definition

Reinforcement Learning

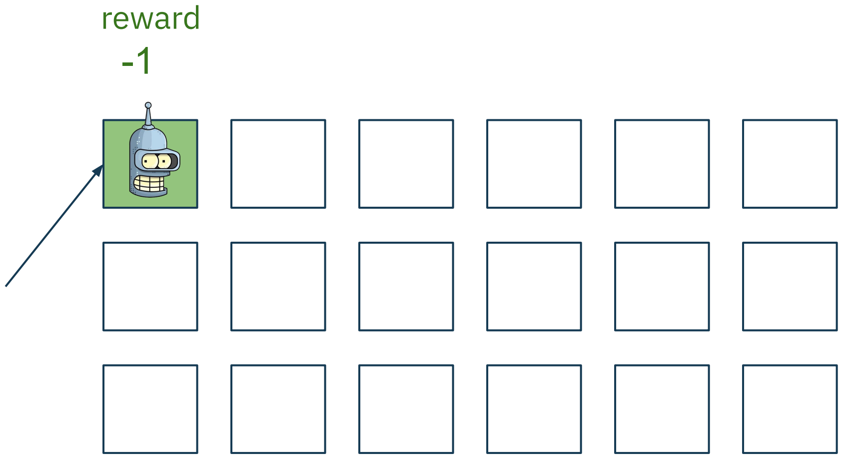
technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



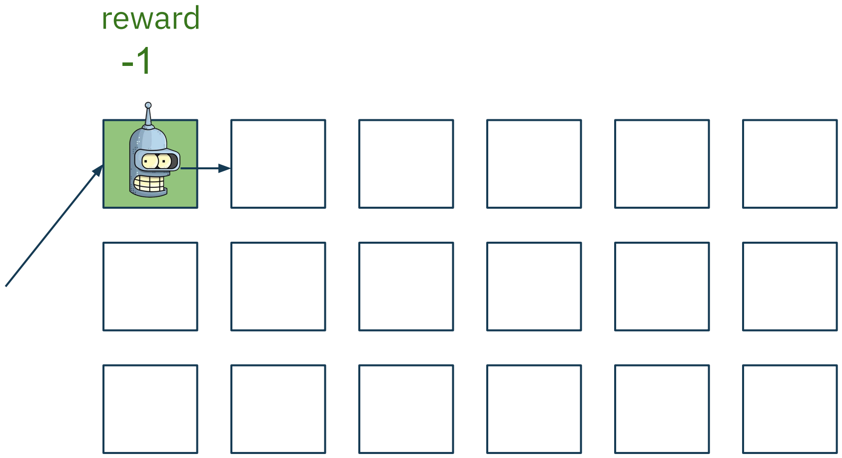
A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

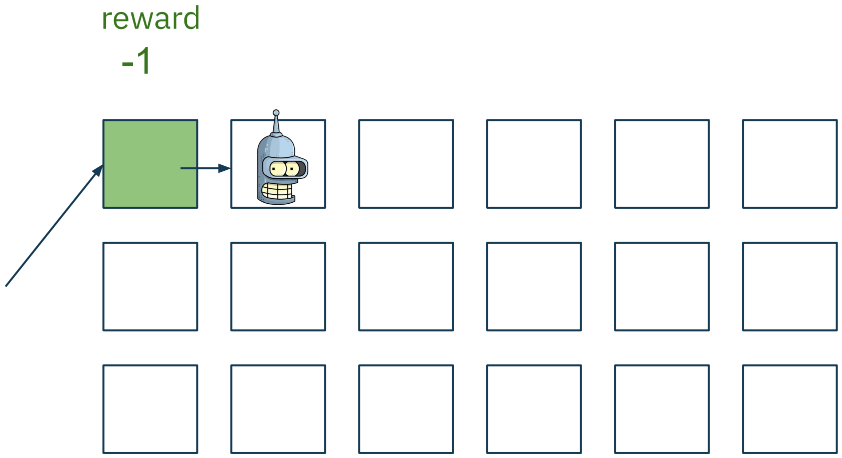
Cumulative Reward

-1



A definition

Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



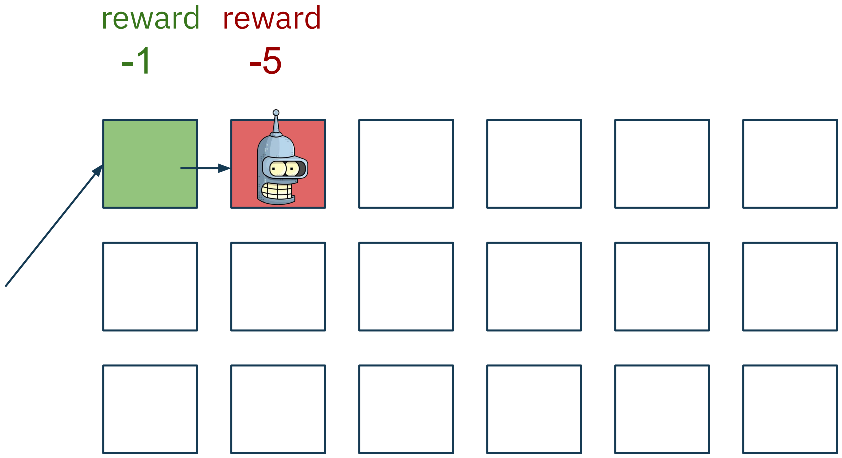
A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

Cumulative Reward

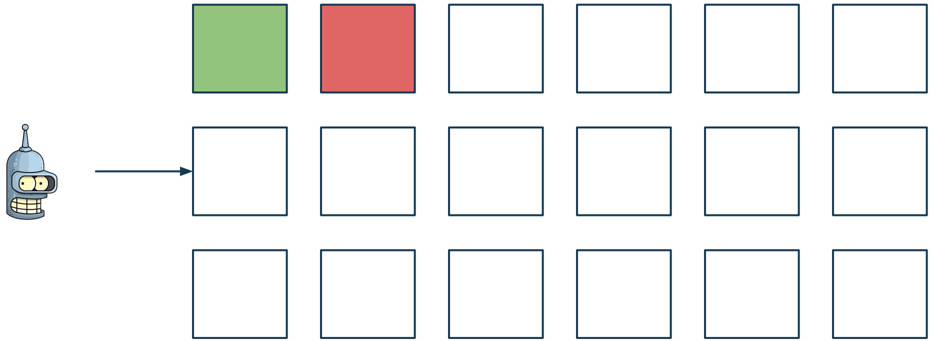
-6



A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

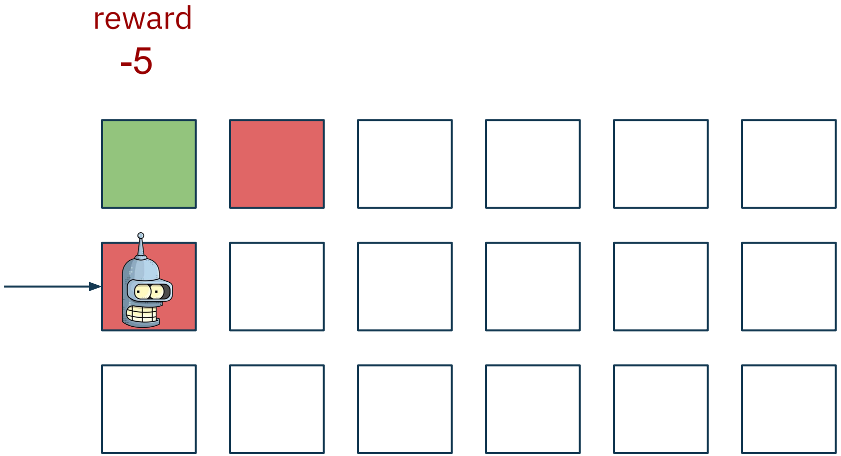


A definition

Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

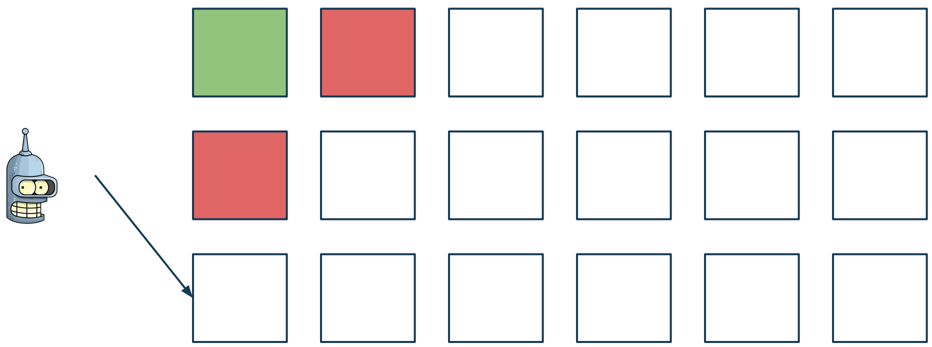
Cumulative Reward

-5



A definition

Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**



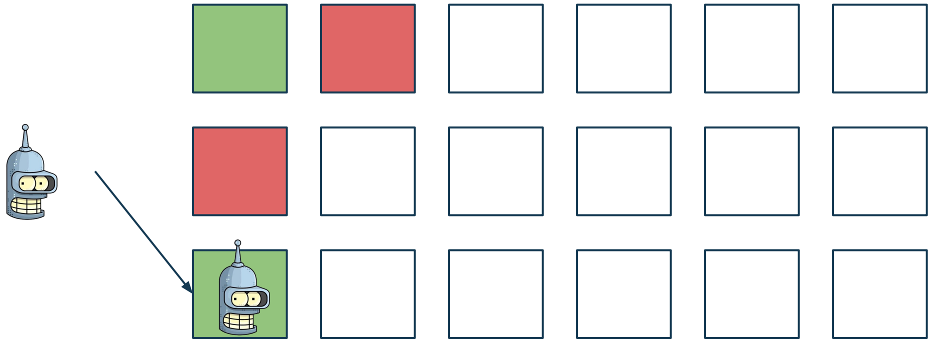
A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

Cumulative Reward

-1

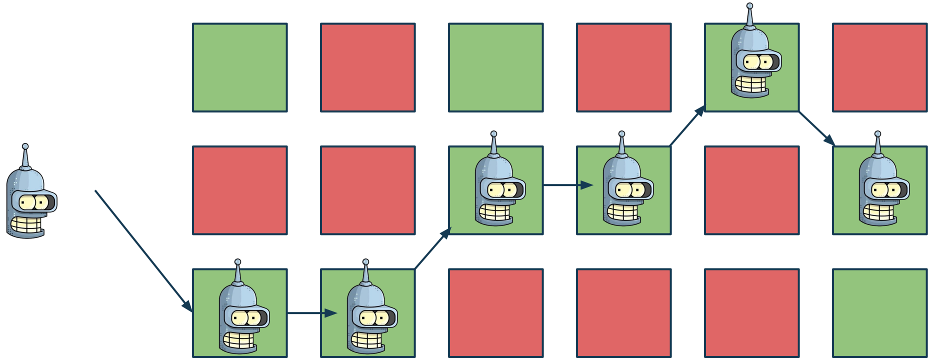


A definition

Reinforcement Learning	technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

Cumulative Reward

-6



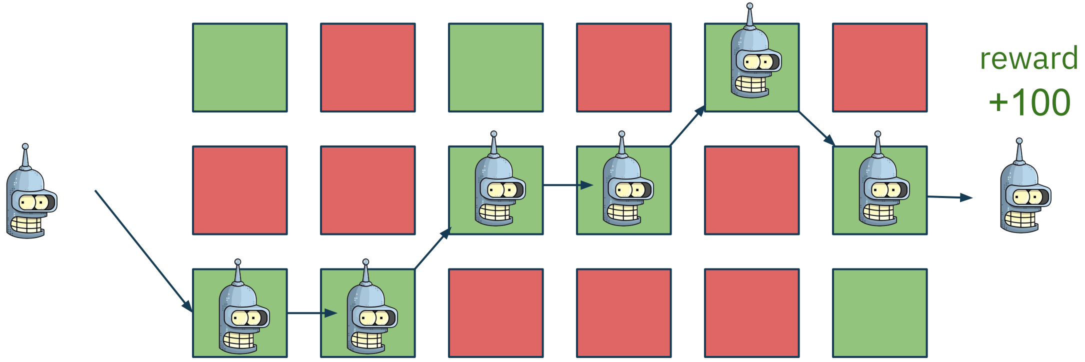
A definition

Reinforcement Learning

technique that trains **agents** to **map states into actions** to **maximize a cumulative reward**

Cumulative Reward

94



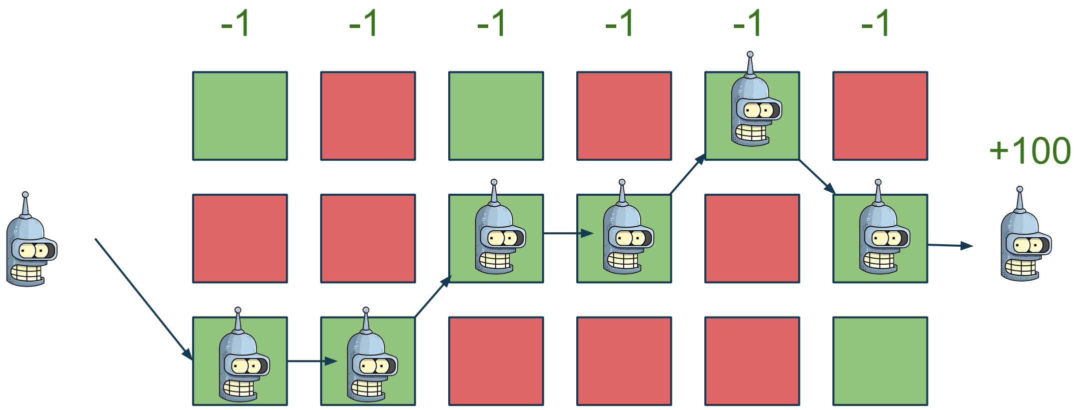
On reward

Goal and reward coherence

we want the agent goes as fast as possible from A to B. We need to choose an appropriate reward signal!

Cumulative Reward

94



On reward

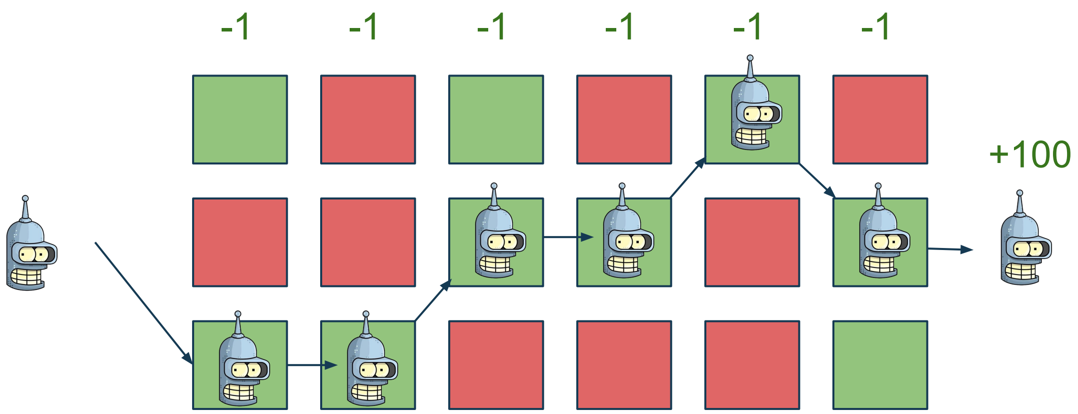
Goal and reward coherence

we want the agent goes as fast as possible from A to B. We need to choose an appropriate reward signal!

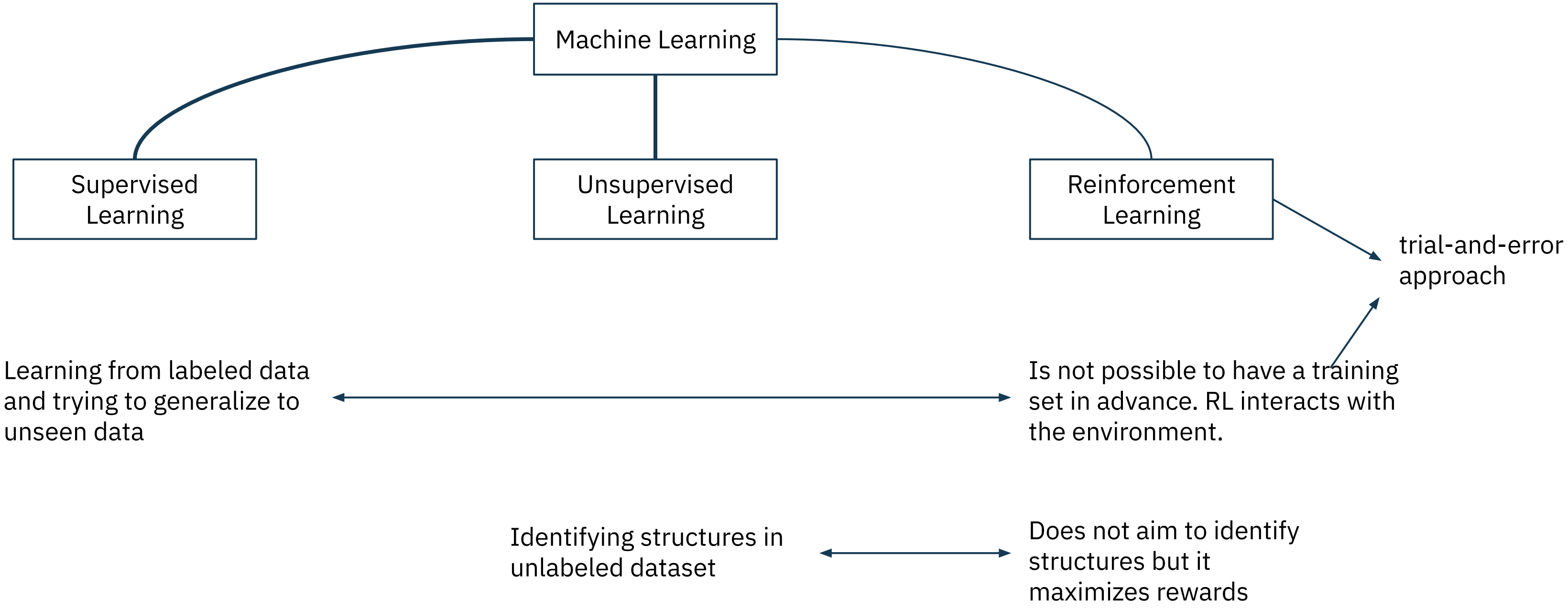
Cumulative Reward

94

Why negative values?



A definition



Elements of RL

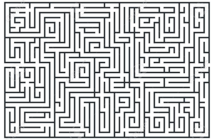
Mathematical definition

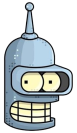
List of the ingredients

Elements of Reinforcement Learning

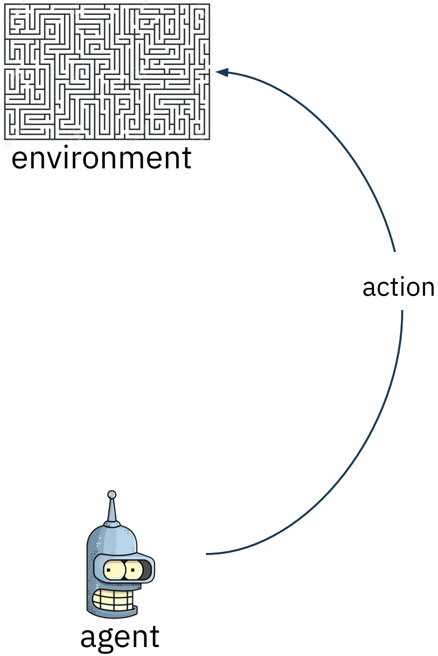
- **Environment** — usually exposes uncertainty
- **Agent** — is a decision-maker and has a goal
- **Policy (of the agent)** — defines the agent behaviour
- **Reward signal** — defines the goal of the agent
- **Model of the environment** — exists model-free and model-based RL algorithms
- **Value function (of a policy)** — quantifies how good is for an agent to be in a specific state (if it follows that policy)

Observability

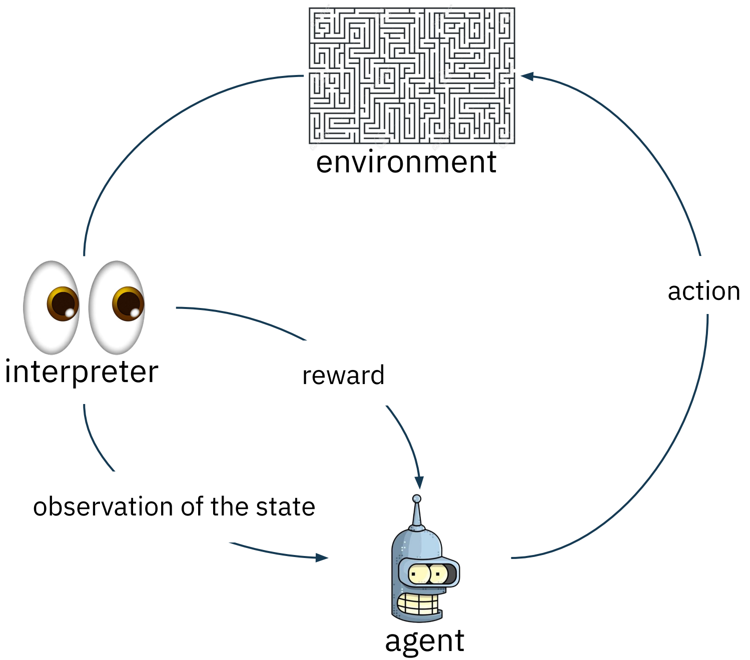

environment


agent

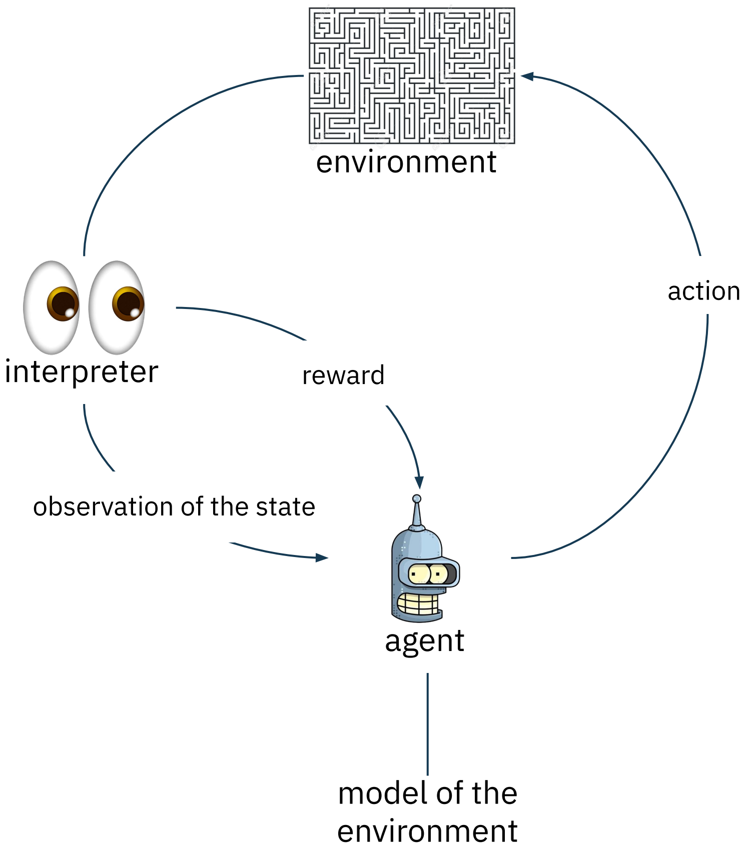
Observability



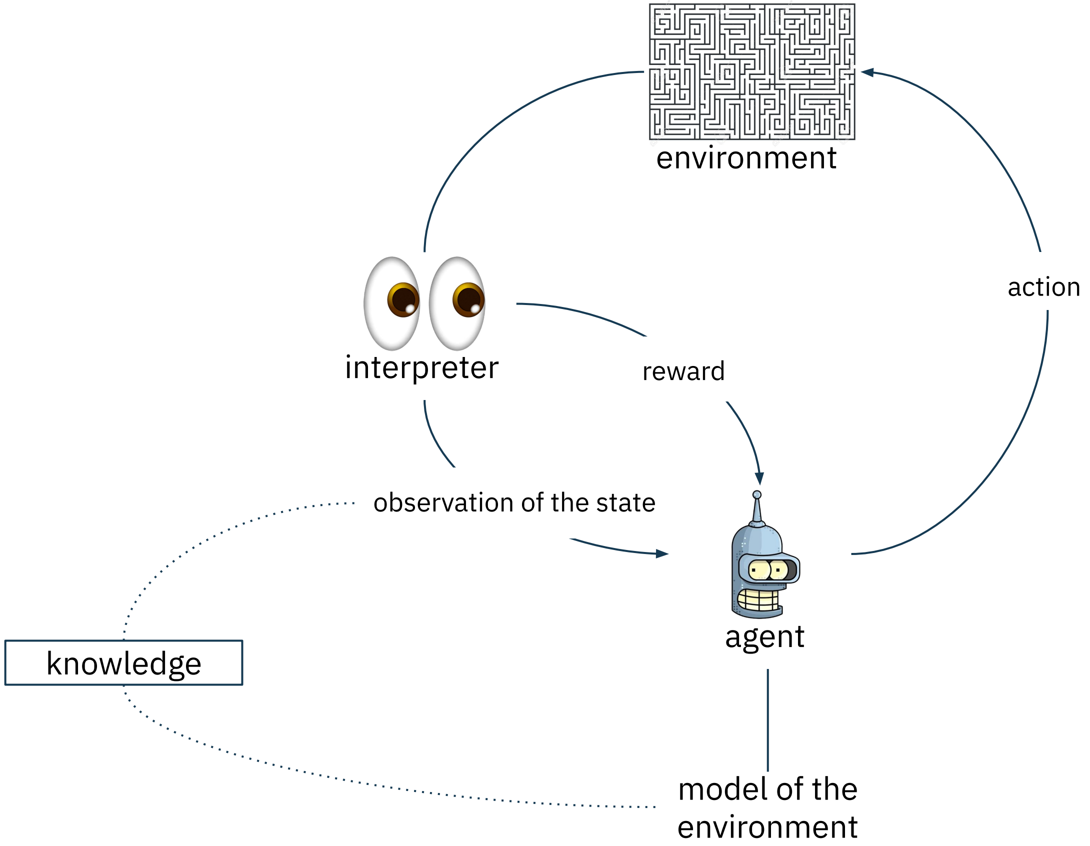
Observability



Observability



Observability



Observability

RL = learning + prediction + controlling

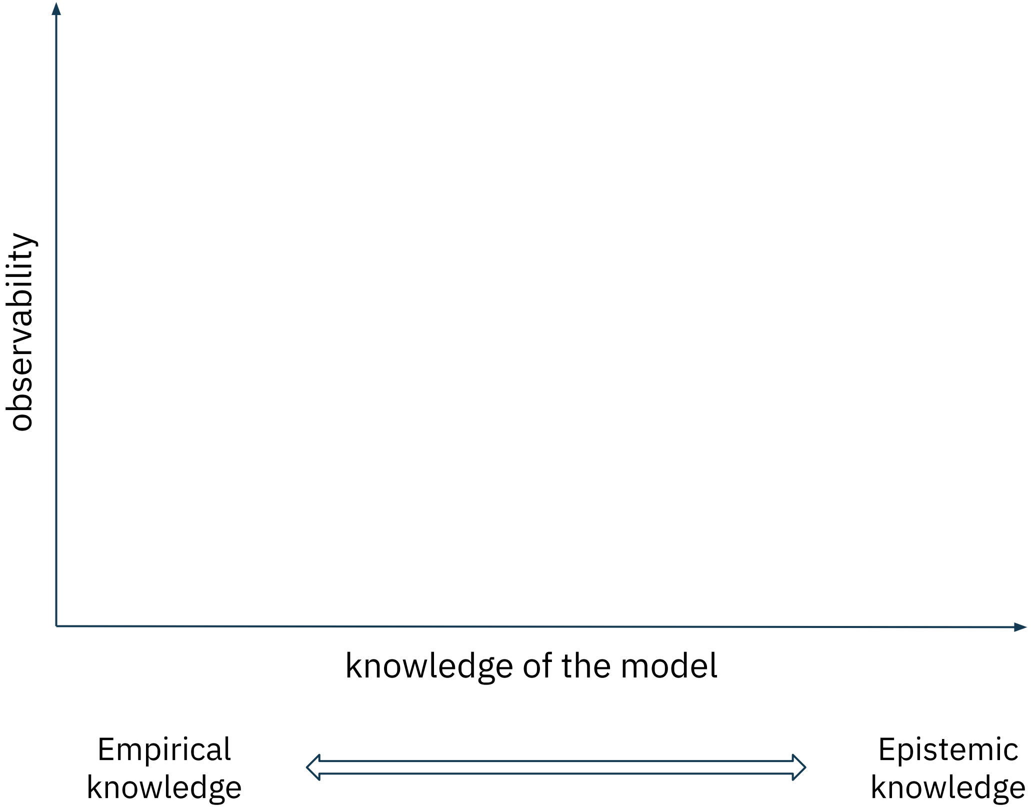
Building a model of the environment

Knowing the cumulative reward I'll get following a policy

Discovering the best action

Knowledge of the environment

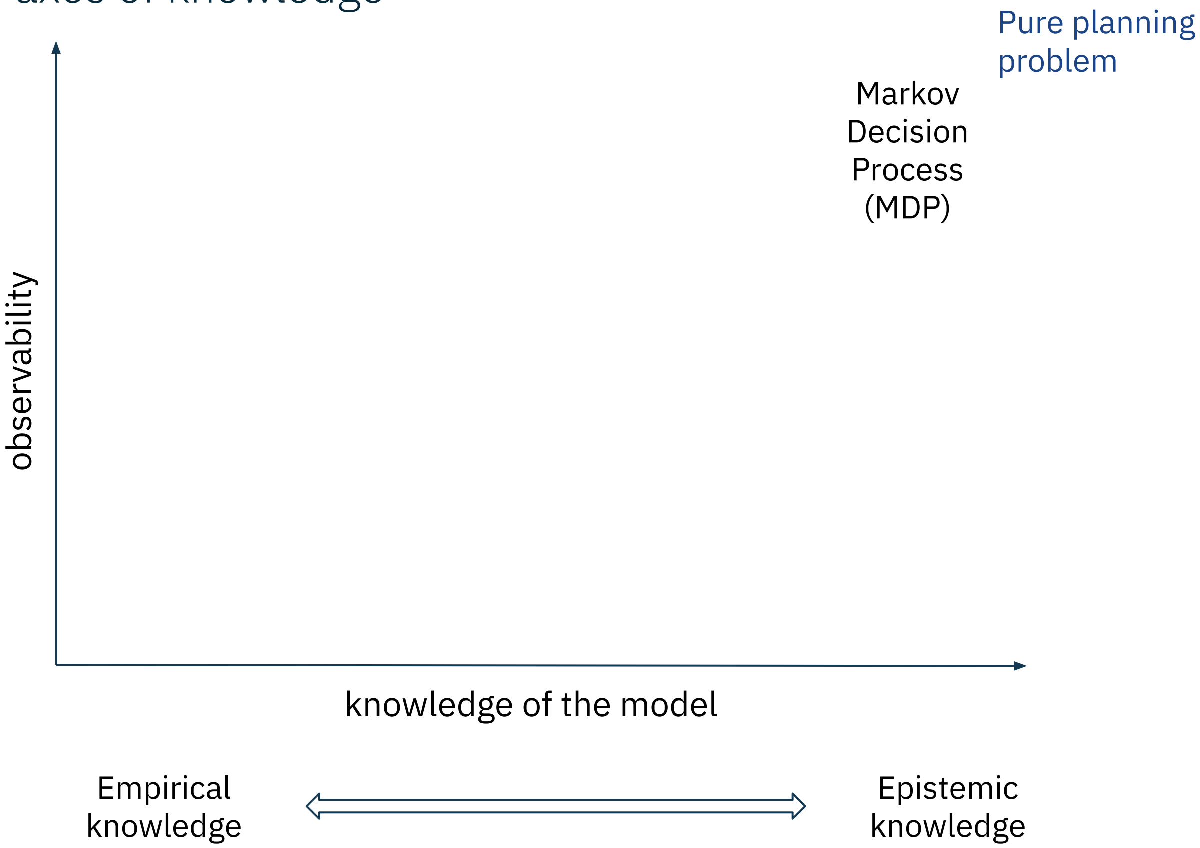
The two axes of knowledge



Knowledge of the environment

The two axes of knowledge

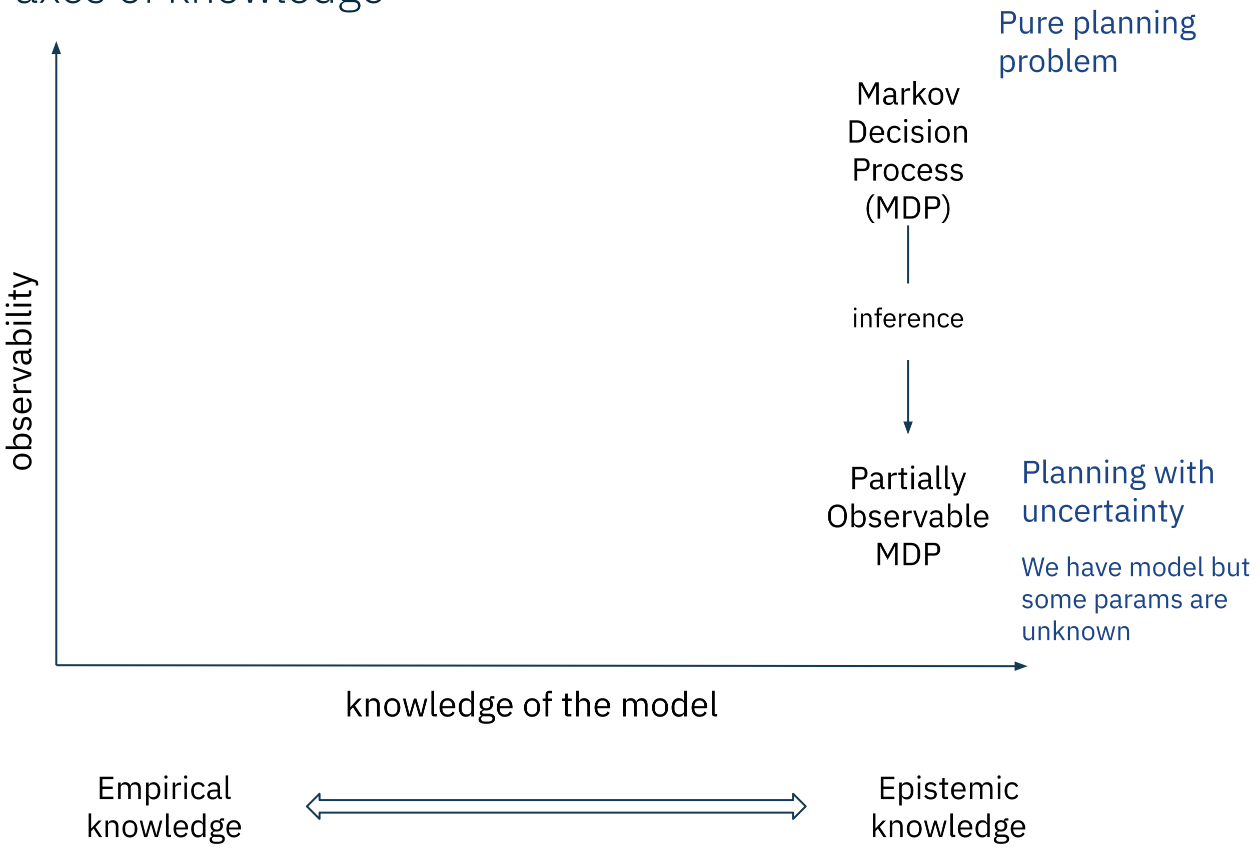
<u>Markovian process</u>
only matter knowledge
of the actual state



Knowledge of the environment

The two axes of knowledge

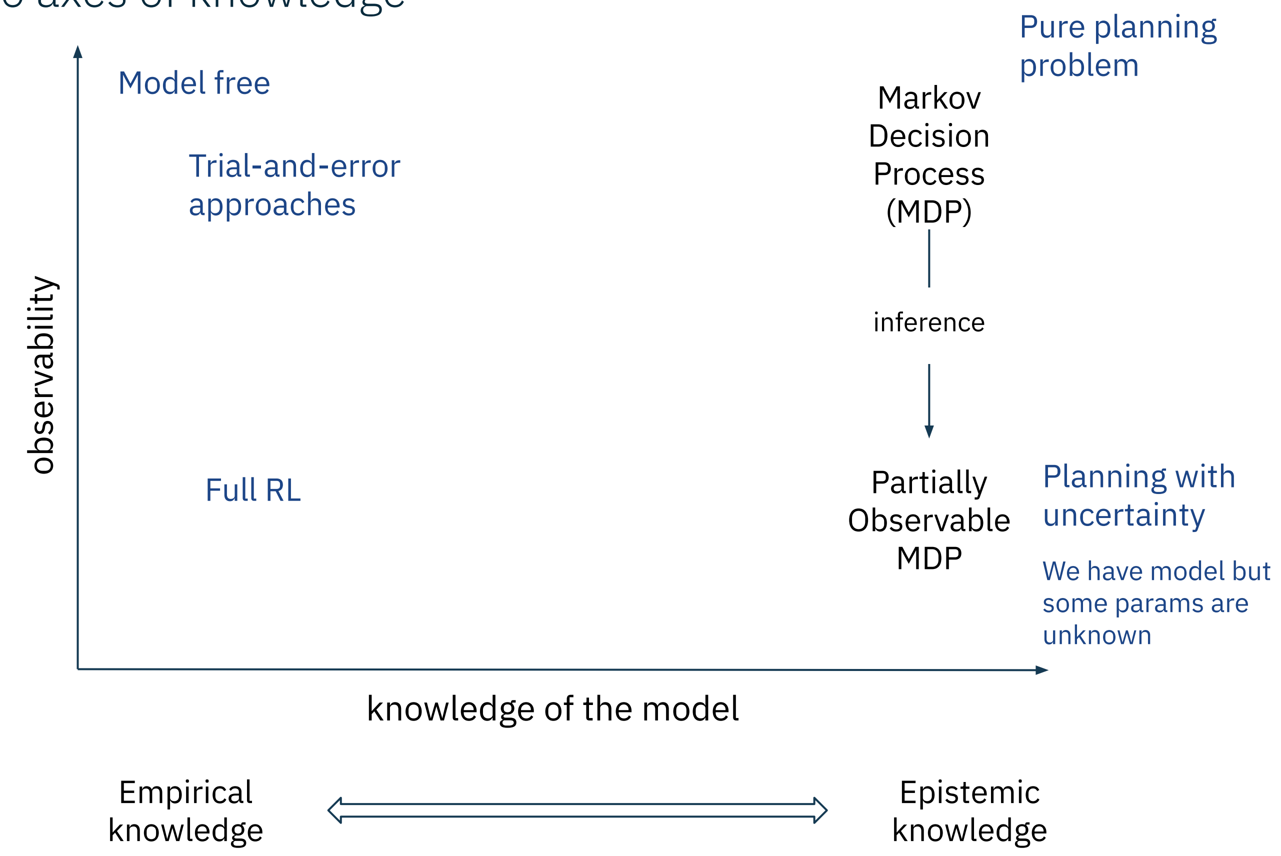
<u>Markovian process</u>
only matter knowledge of the actual state



Knowledge of the environment

The two axes of knowledge

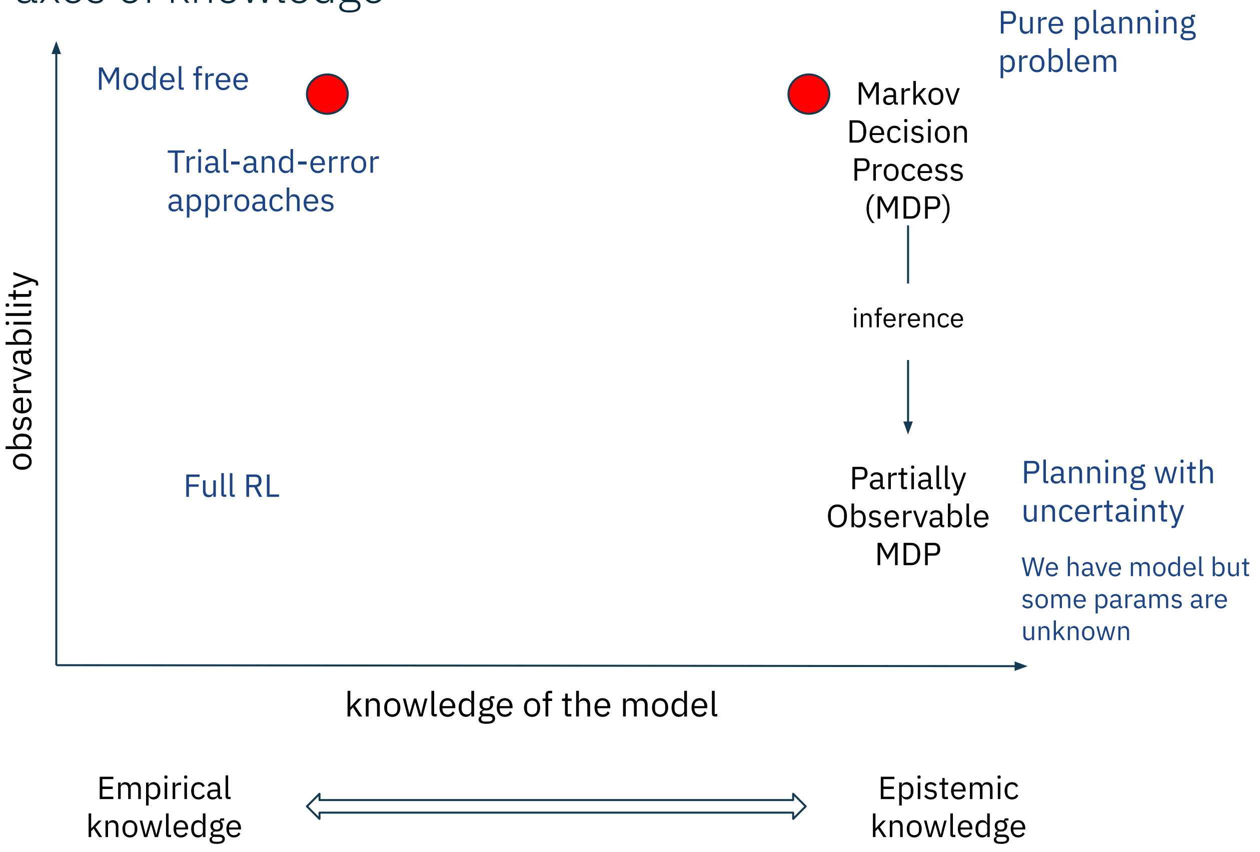
<u>Markovian process</u>
only matter knowledge of the actual state



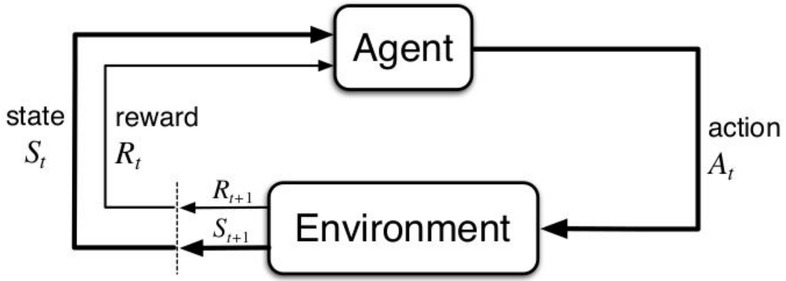
Knowledge of the environment

Markovian process
only matter knowledge of the actual state

The two axes of knowledge



(finite) Markov Decision Process



trajectory

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

dynamics

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

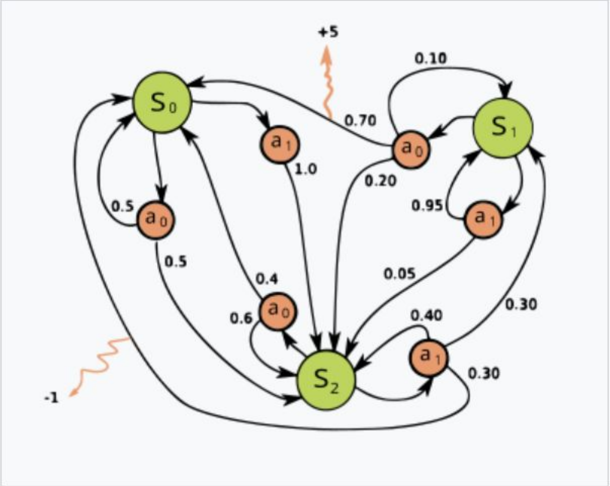
Perfect knowledge
of the model

(finite) Markov Decision Process

Definition [edit]

A Markov decision process is a 4-tuple (S, A, P_a, R_a), where:

- S is a set of states called the *state space*. The state space may be discrete or continuous, like the set of real numbers.
- A is a set of actions called the *action space* (alternatively, A_s is the set of actions available from state s). As for state, this set may be discrete or continuous.
- $P_a(s, s')$ is, on an intuitive level, the probability that action a in state s at time t will lead to state s' at time $t + 1$. In general, this probability transition is defined to satisfy $\Pr(s_{t+1} \in S' \mid s_t = s, a_t = a) = \int_{S'} P_a(s, s')ds'$, for every $S' \subseteq S$ measurable. In case the state space is discrete, the integral is intended with respect to the counting measure, so that the latter simplifies as $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$; In case $S \subseteq \mathbb{R}^d$, the integral is usually intended with respect to the Lebesgue measure.
- $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a.



Example of a simple MDP with three states (green circles) and two actions (orange circles), with two rewards (orange arrows)

A policy function π is a (potentially probabilistic) mapping from state space (S) to action space (A).

(finite) Markov Decision Process

trajectory

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$$

dynamics

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

state-transition probability

$$p(s' \mid s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

expected reward (I)

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

expected reward (II)

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

Reward signal

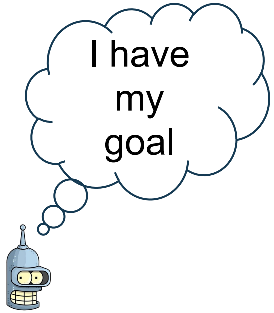
Reward hypothesis: that all of what we mean by <u>goals and purposes</u> can be well thought of as the <u>maximization of the expected value of the cumulative sum of a received scalar signal (called reward).</u>

Reward R_{t+1}

Return $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$

Discounted Return $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Reward signal



Reward hypothesis: that all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

Short-term view

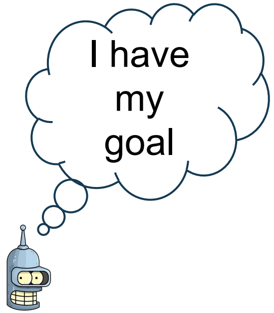
Reward R_{t+1}

Long-term view

Return $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$

Discounted Return $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Reward signal



Reward hypothesis: that all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

Short-term view

Reward: R_{t+1}

Long-term view

Return: $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$

Discounted Return: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

RECURSIVE DEFINITION

Policy

Policy

is a mapping from states to probabilities of selecting each possible action

$$\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$$

If we are at time t, $\pi(a|s)$ is the probability of having $A_t = a \wedge S_t = s$

can be *deterministic*

Value function

Value Function

is a function that quantify how good is to be on a state and follows a specific policy

$$v_\pi : \mathcal{S} \to \mathbb{R}$$

state-value function

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

action-value function

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

Solving a RL problem

find a policy that achieves the maximum reward over the long run

optimal policy

$\pi_* \succeq \pi \quad \forall \pi \in$ policies

Solving a RL problem

find a policy that achieves the maximum reward over the long run

optimal policy

$\pi_* \succeq \pi \quad \forall \pi \in$ policies

$$\pi' \succeq \pi \iff \forall s \in \mathcal{S},\ v_{\pi'}(s) \geq v_{\pi}(s)$$

Solving a RL problem

find a policy that achieves the maximum reward over the long run

optimal policy

$\pi_* \succeq \pi \quad \forall \pi \in$ policies

$$\pi' \succeq \pi \iff \forall s \in \mathcal{S},\ v_{\pi'}(s) \geq v_\pi(s)$$

optimal state-value function

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

optimal action-value function

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

Solving a RL problem

find a policy that achieves the maximum reward over the long run

optimal policy

$\pi_* \succeq \pi \quad \forall \pi \in$ policies

$$\pi' \succeq \pi \iff \forall s \in \mathcal{S},\ v_{\pi'}(s) \geq v_\pi(s)$$

optimal state-value function

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

optimal action-value function

$$q_*(s, a) \doteq \max_\pi q_\pi(s, a)$$

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

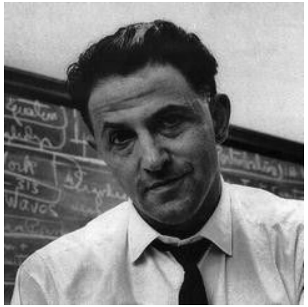
Dynamic Programming

How to solve MDP problems

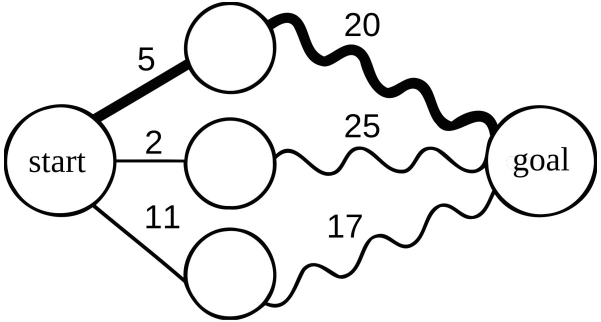
Dynamic Programming

Mr. Richard Ernest Bellman

Algorithm paradigm useful to solve a specific class of problems that can be decomposed in sub-problems in recursive way

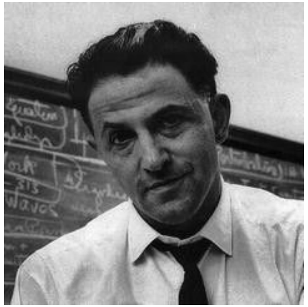


Bellman, 1950s



Dynamic Programming

In the RL context



Bellman, 1950s

Collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a MDP.

Key idea: use value function to organize and structure the search of optimal policies

Consistency relation of state-value function

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}.
\end{aligned}
$$

Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

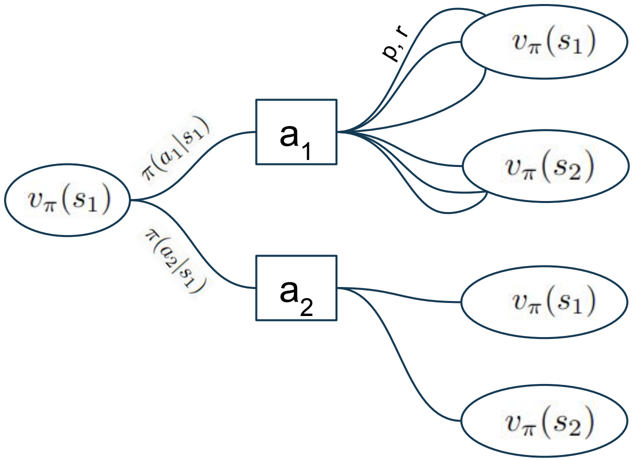
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

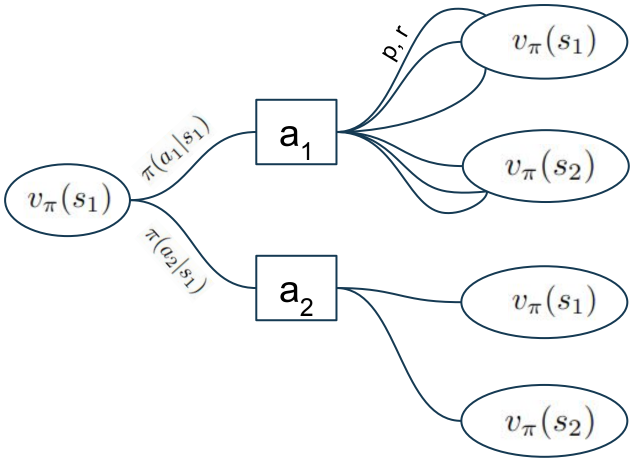


Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

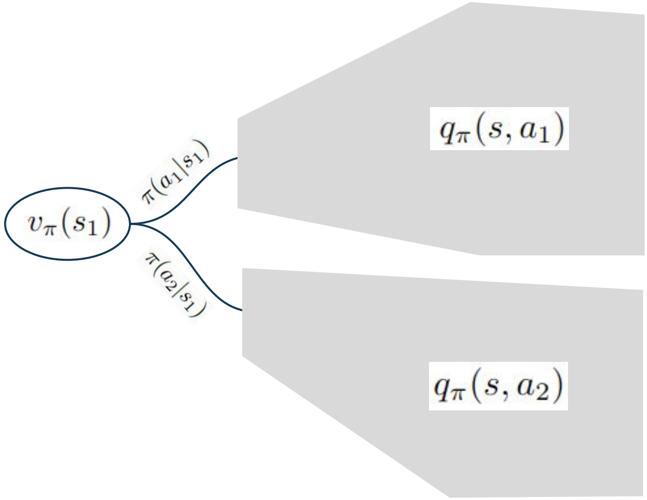


Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

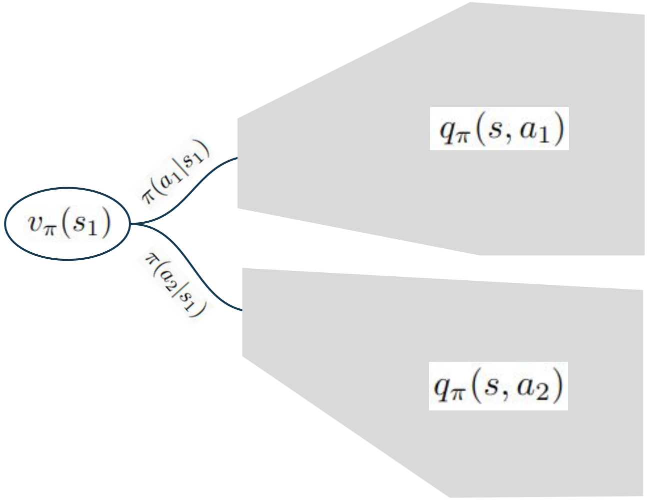


Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$



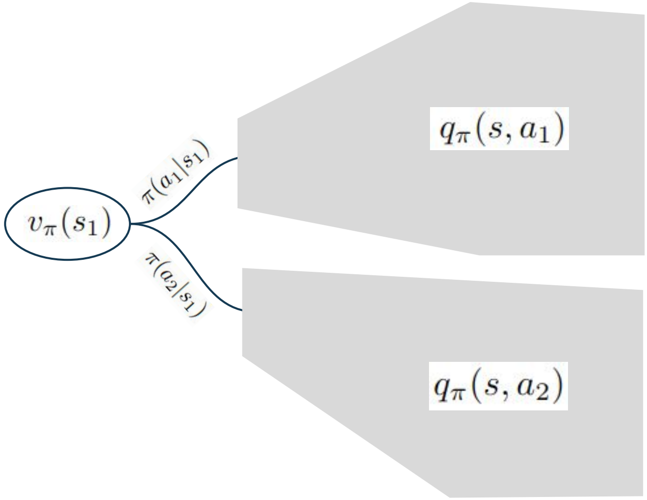
Dynamic Programming

Towards the Bellman Equation

What about the optimal policy and the optimal state-value function?

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$



Dynamic Programming

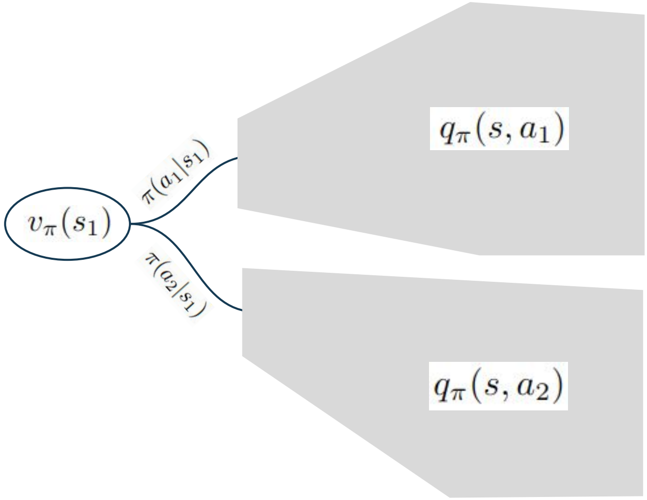
Towards the Bellman Equation

What about the optimal policy and the optimal state-value function?

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

It's an average



Dynamic Programming

Towards the Bellman Equation

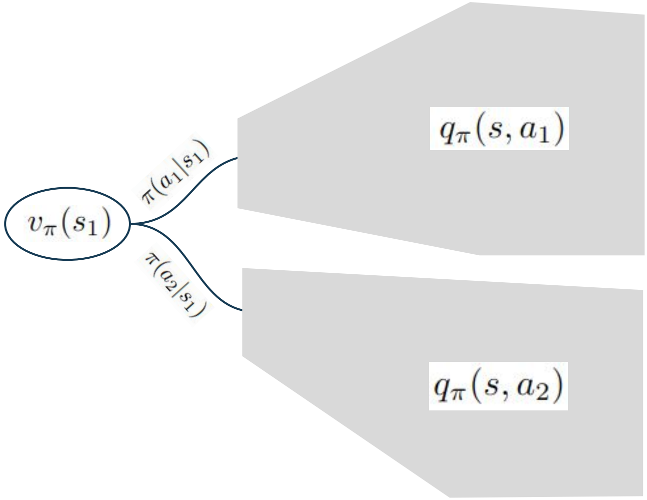
What about the optimal policy and the optimal state-value function?

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

It's an average

The optimal policy is a policy so it should satisfy the consistency relation



Dynamic Programming

Towards the Bellman Equation

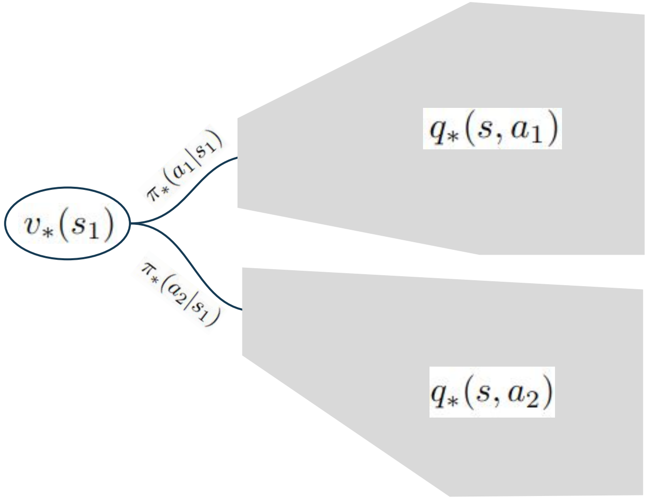
Consistency relation of state-value function

$$v_*(s) = \sum_a \pi_*(a|s) q_*(s,a)$$

What about the optimal policy and the optimal state-value function?

It's an average

The optimal policy is a policy so it should satisfy the consistency relation

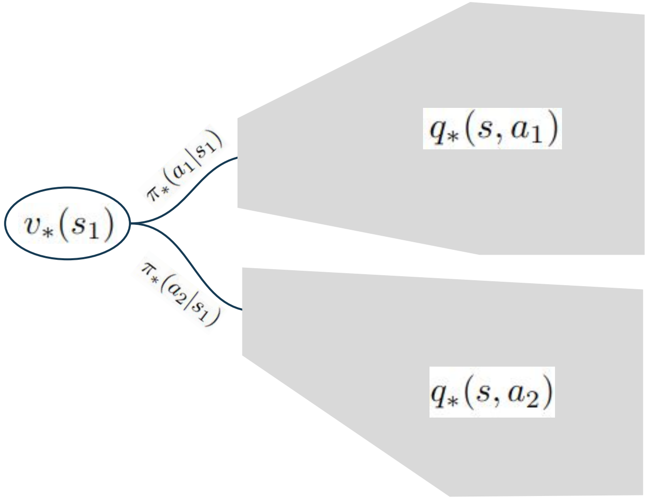


Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_*(s) = \sum_a \pi_*(a|s) q_*(s,a)$$



What about the optimal policy and the optimal state-value function?

It's an average

The optimal policy is a policy so it should satisfy the consistency relation

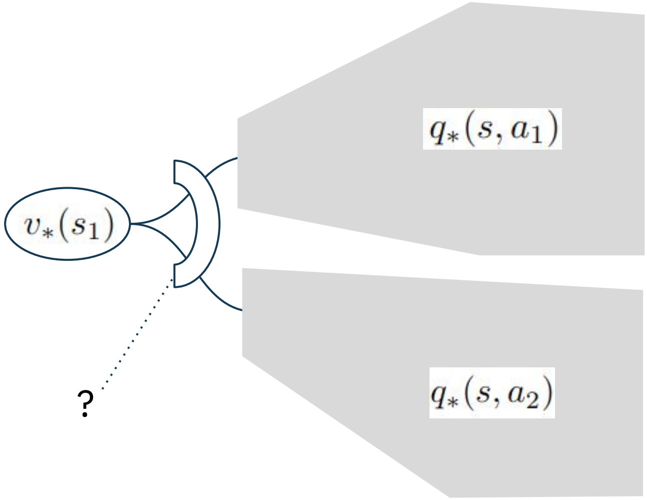
The optimal policy is *optimal*

Dynamic Programming

Towards the Bellman Equation

Consistency relation of state-value function

$$v_*(s) = \sum_a \pi_*(a|s) q_*(s,a)$$



What about the optimal policy and the optimal state-value function?

It's an average

The optimal policy is a policy so it should satisfy the consistency relation

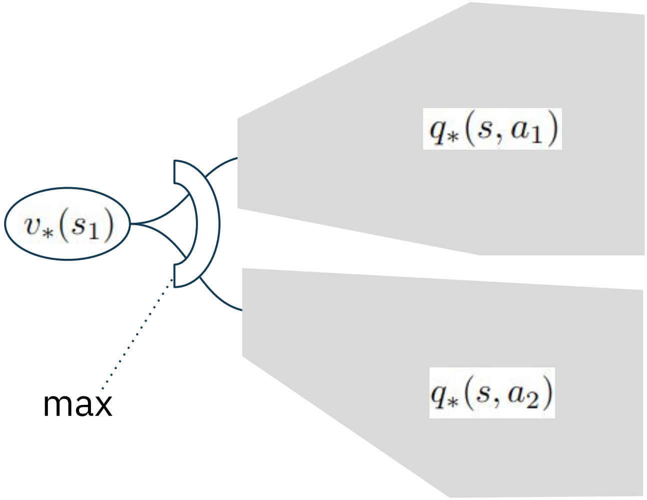
The optimal policy is *optimal*

Dynamic Programming

Bellman Equation

Bellman equation

$$v_*(s) = \max_a q_*(s, a)$$



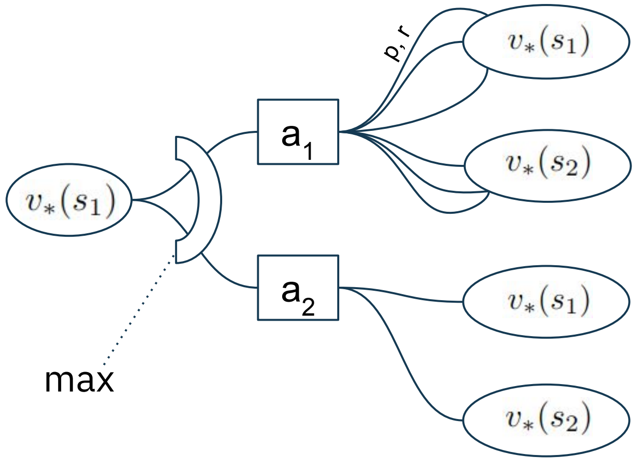
Dynamic Programming

Bellman Equation

Bellman equation

$$v_*(s) = \max_a q_*(s, a)$$

$$v_*(s) = \max_a \sum_{s', r} p(s', r|s, a)[r + \gamma v_*(s')]$$



Dynamic Programming

Bellman Equation

Bellman equation

$$v_*(s) = \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]$$

$$q_*(s, a) = \sum_{s',r} p(s', r | s, a)[r + \gamma \max_{a} q_*(s', a')]$$

Dynamic Programming

How to find the optimal policy?

$$\pi \xrightarrow{\text{Iterative procedure}} \pi^*$$

Dynamic Programming

How to find the optimal policy?

Consistency relation of the state-value function

Policy evaluation

$$\pi \longrightarrow v_\pi \longrightarrow \pi'$$

Dynamic Programming

How to find the optimal policy?

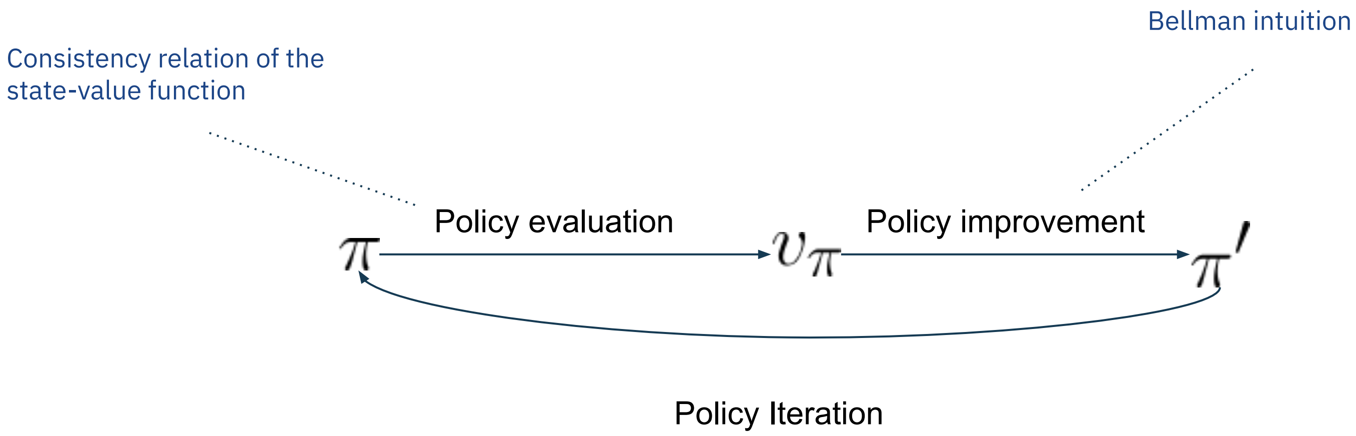
Consistency relation of the state-value function

Bellman intuition

π —Policy evaluation→ v_π —Policy improvement→ π'

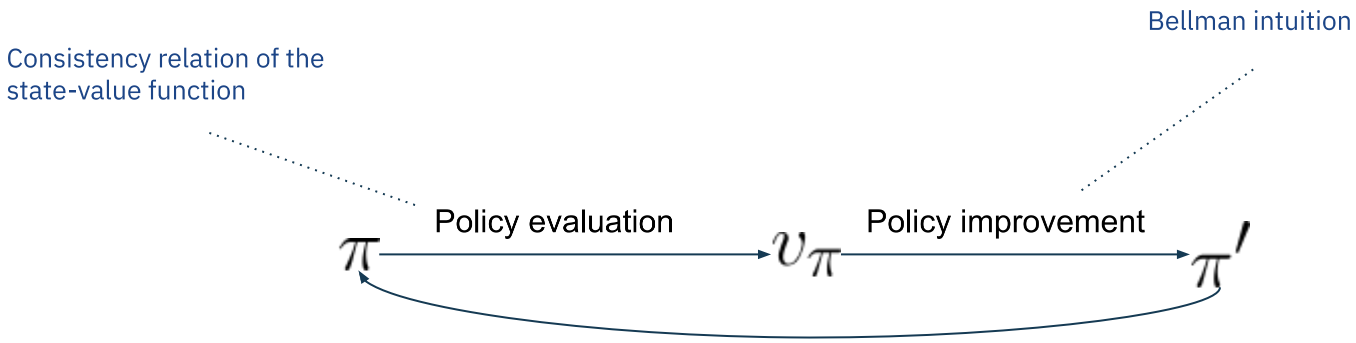
Dynamic Programming

How to find the optimal policy?



Dynamic Programming

How to find the optimal policy?



Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Does it converge? Yes

Dynamic Programming

Policy evaluation

$$\pi \xrightarrow{\text{Policy evaluation}} v_\pi$$

Consistency relation of state-value function

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

Dynamic Programming

Policy evaluation

$\pi \xrightarrow{\text{Policy evaluation}} v_\pi$

Consistency relation of state-value function	$v_\pi(s) = \sum_a \pi(a\|s) \sum_{s',r} p(s',r\|s,a)\left[r + \gamma v_\pi(s')\right]$
Iterative policy evaluation	$v_{k+1}(s) = \sum_a \pi(a\|s) \sum_{s',r} p(s',r\|s,a)[r + \gamma v_k(s')]$

Dynamic Programming

Policy evaluation

$\pi \xrightarrow{\text{Policy evaluation}} v_\pi$

Consistency relation of state-value function

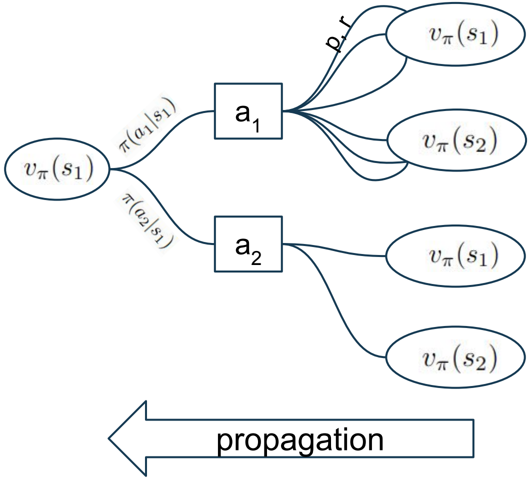
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

Iterative policy evaluation

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$$

<u>2 ways of updating</u>: *in-place* vs *two arrays version*

Faster, depends on ordering of update



Policy Evaluation

Algorithm

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Policy Evaluation

Algorithm

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$ consistency relation
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Policy Evaluation

Algorithm

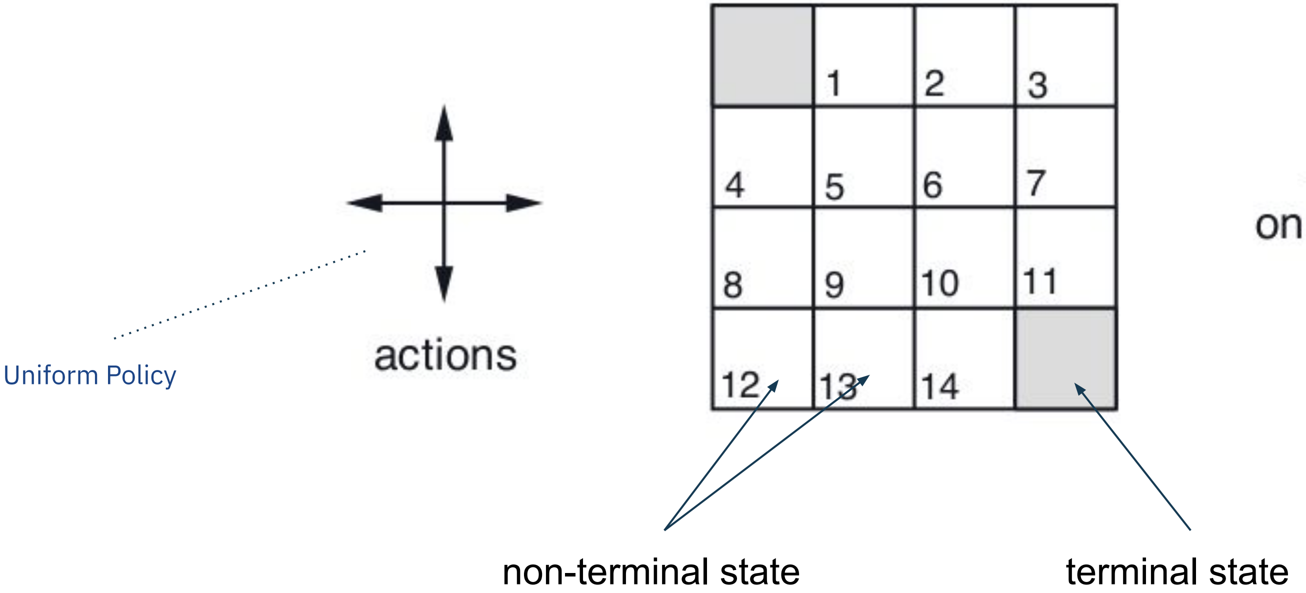
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a) \big[r + \gamma V(s') \big]$ — consistency relation
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ — Stability of state-value function

Policy Evaluation

Example



$$R_t = -1$$

on all transitions

Policy Evaluation

Example - 1st iteration

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

0.0			
			0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - 1st iteration

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

⇒

0.0			
			0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[\underbrace{r}_{-1} + \underbrace{\gamma v_\pi(s')}_{0}]$$

-1 0

Policy Evaluation

Example - 1st iteration

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - 1st iteration

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

0.0	?		
?			
			?
		?	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - 2nd iteration

← -1/3 +

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

⇒

0.0	?		
?			
			?
		?	0.0

$$v_\pi(s) = \sum_a \underbrace{\pi(a|s)}_{1/3} \sum_{s',r} \underbrace{p(s',r|s,a)}_{1}[\underbrace{r}_{-1} + \gamma \underbrace{v_\pi(s')}_{0}]$$

Policy Evaluation

Example - 2nd iteration

← -1/3 +

↓ -2/3 +

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

⇒

0.0	?		
?			
			?
		?	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

1/3 1 -1 -1

Policy Evaluation

Example - 2nd iteration

← -1/3 +

↓ -2/3 +

→ -2/3 +

-1.7

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

⇒

0.0	?		
?			
			?
		?	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - 2nd iteration

← -1/3 +

↓ -2/3 +

→ -2/3 +

-1.7

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

⇒

0.0	-1.7		
-1.7			
			-1.7
		-1.7	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - 2nd iteration

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Policy Evaluation

Example - until the end

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

2

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

3

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

∞

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Evaluation

Example - until the end

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

1 ⇒

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

2 ⇒

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

3 ⇒

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

∞

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Evaluation

Example - until the end

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

2

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

3

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

∞

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Improvement

How to find better policies

$v_\pi \xrightarrow{\text{Policy improvement}} \pi'$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy improvement theorem

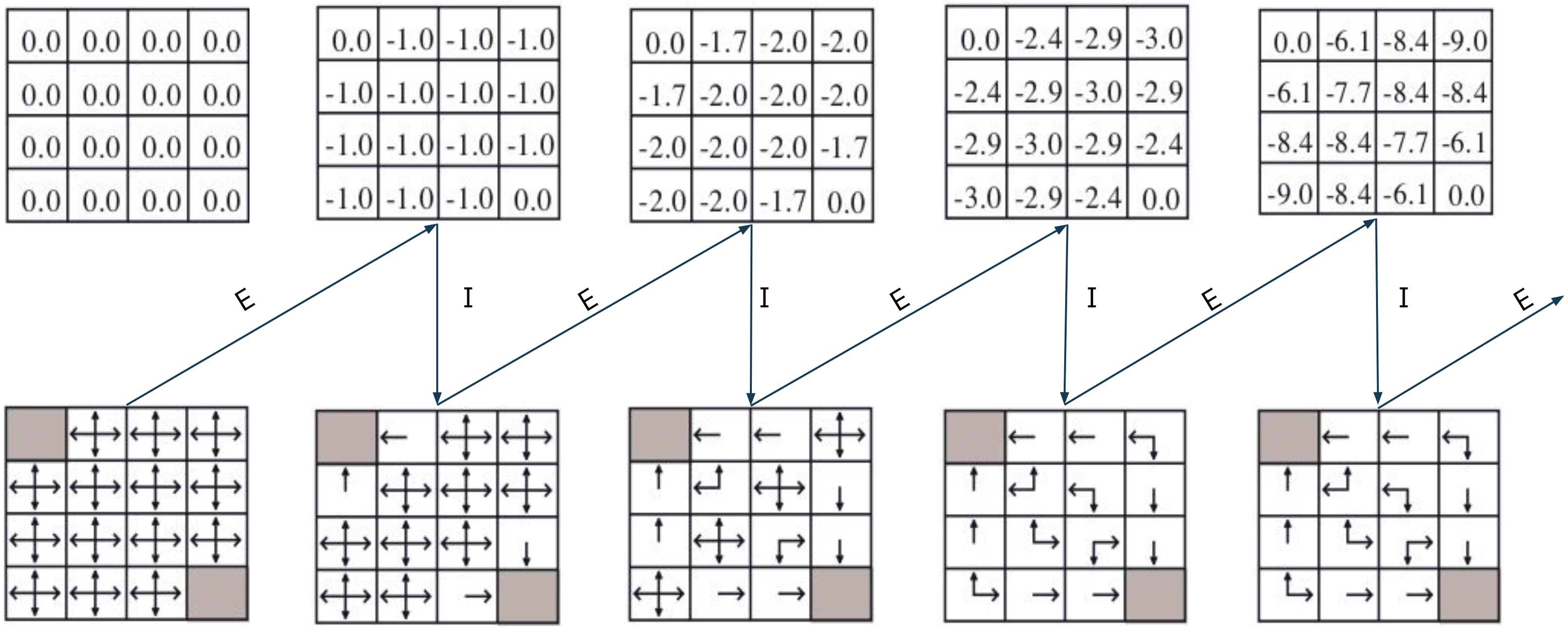
$$q_\pi(s, \pi'(s)) \geq v_\pi(s), \forall s \in \mathcal{S} \Rightarrow v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}$$

Greedy policy approach

$$\begin{aligned} \pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big], \end{aligned}$$

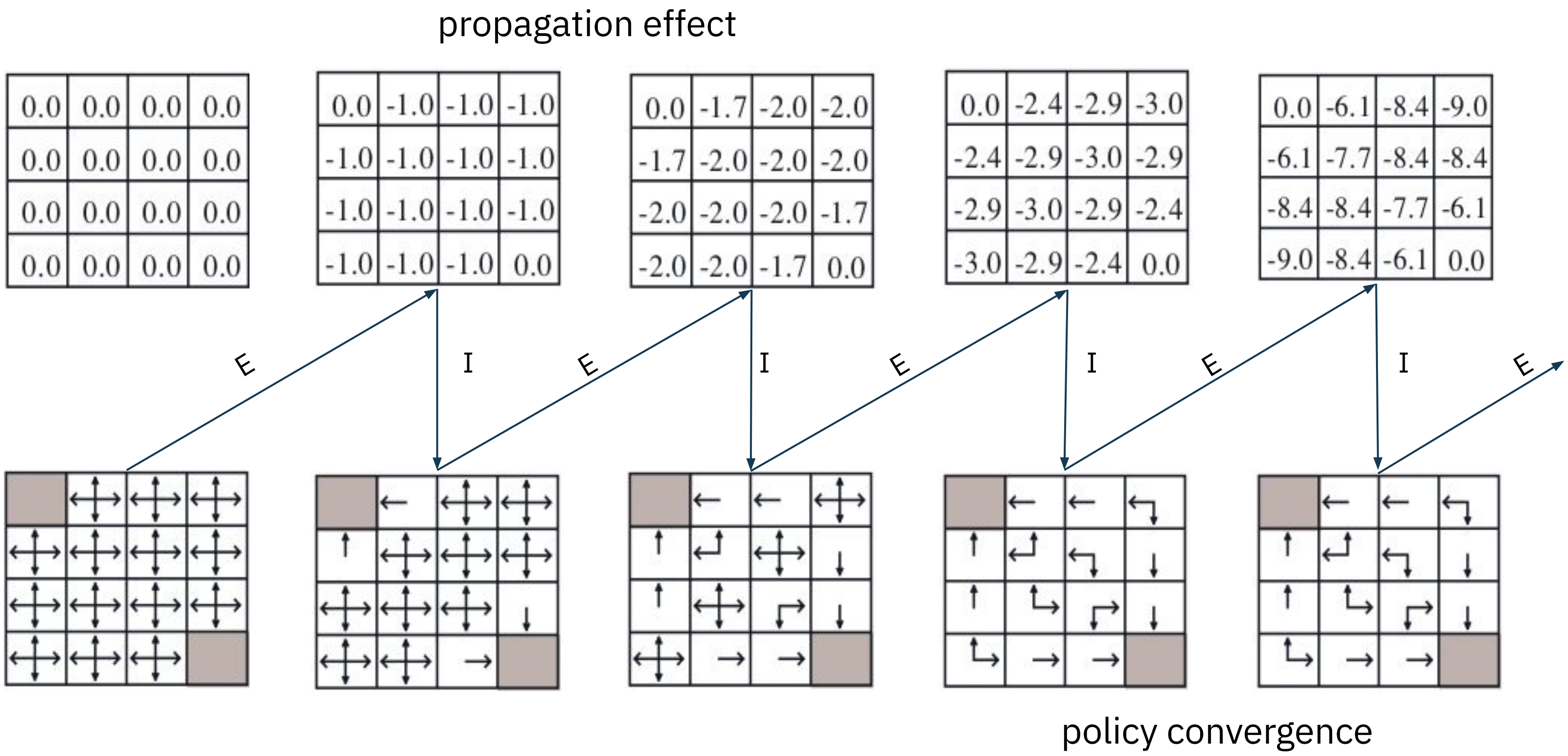
Policy Iteration

Example



Policy Iteration

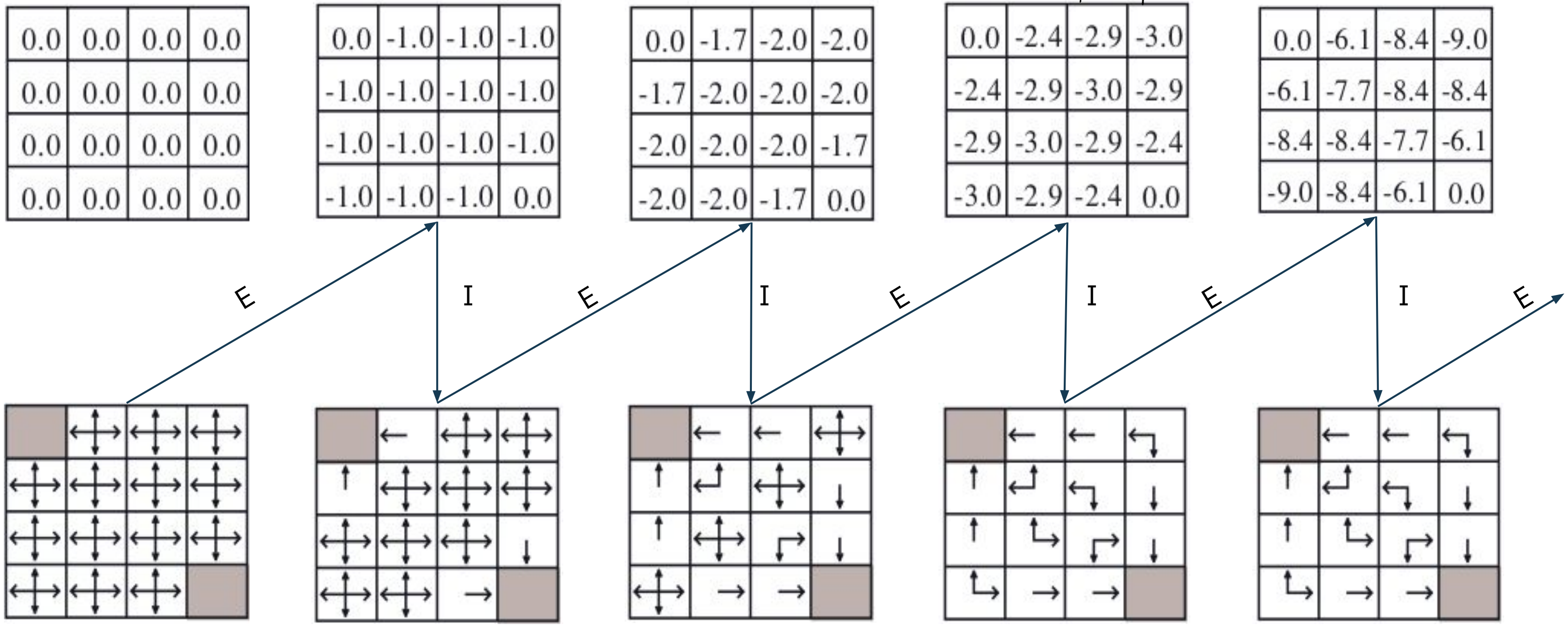
Example



Policy Iteration

Example

propagation effect



policy convergence

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
       If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s') \big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s') \big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

Solving efficiently the Policy Iteration

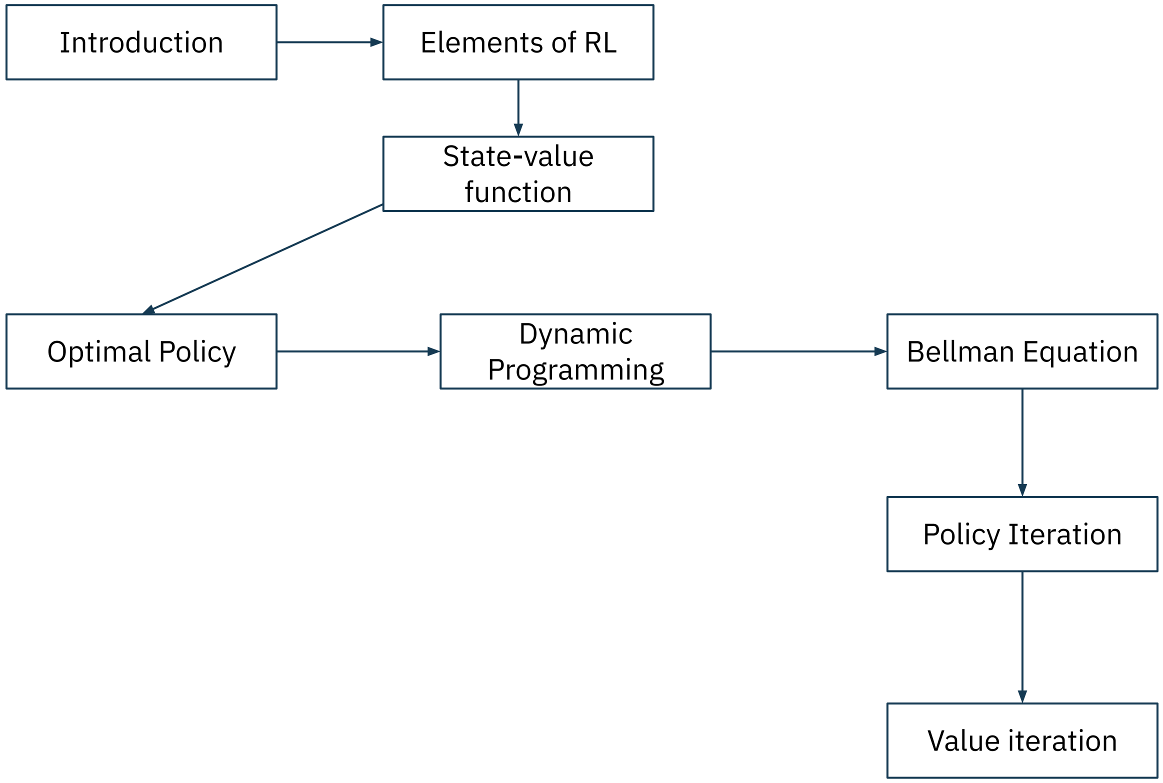
Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Recap

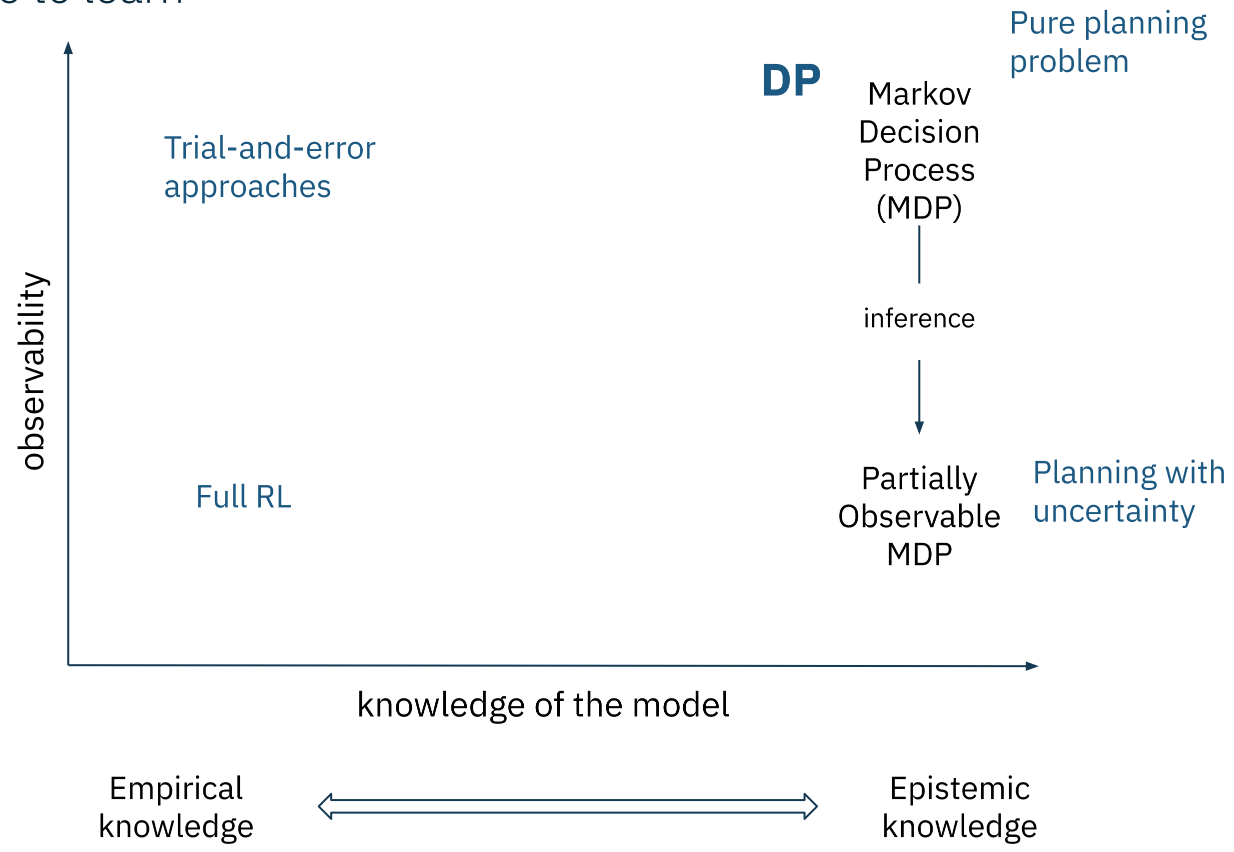

Introduction
Elements of RL
State-value function
Optimal Policy
Dynamic Programming
Bellman Equation
Policy Iteration
Value iteration

Monte Carlo Methods

We are ignorant, we need to learn

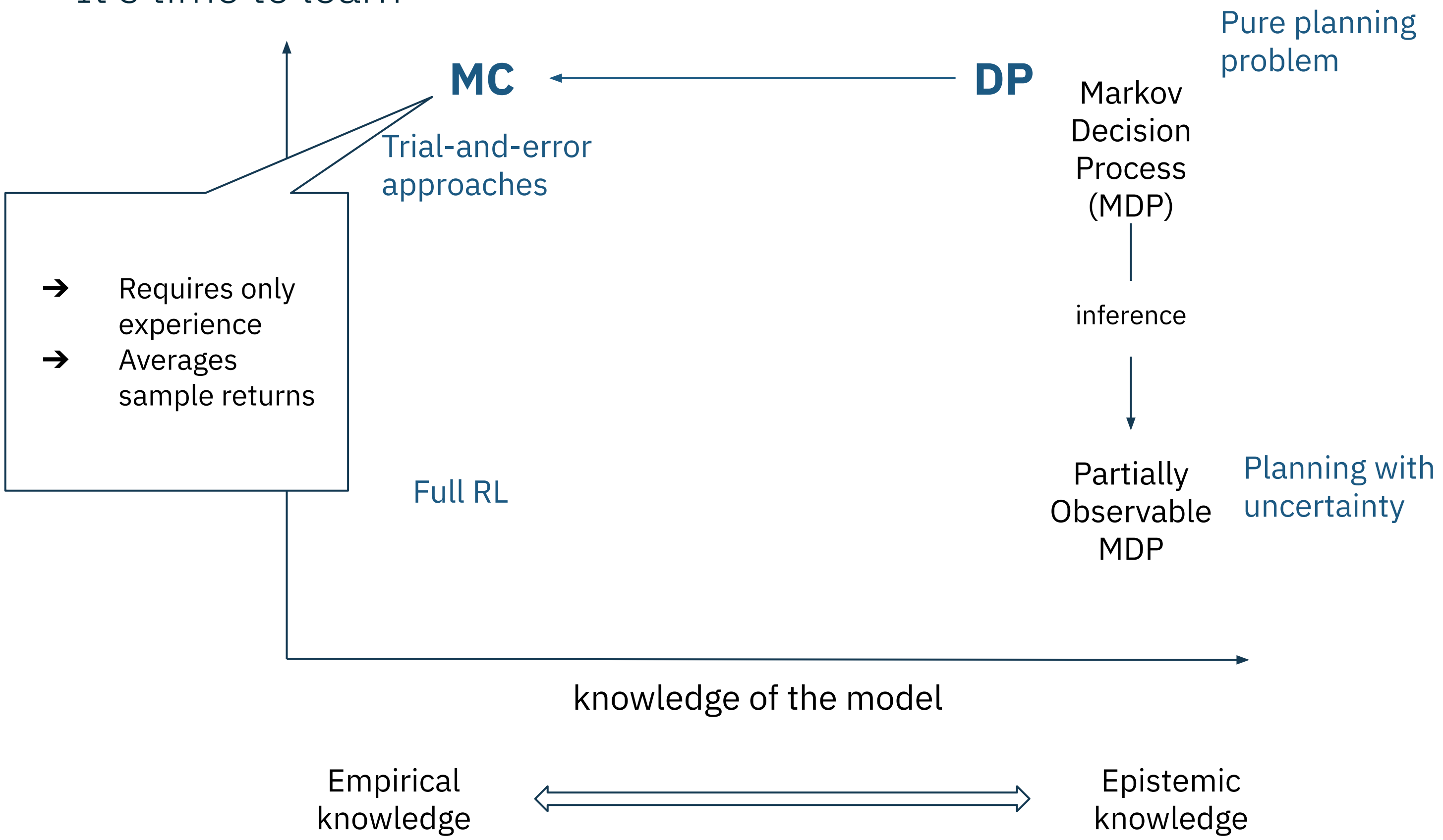
Monte Carlo Methods

It's time to learn



Monte Carlo Methods

It's time to learn



Monte Carlo Methods

First-visit MC prediction idea

Episode 0	$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T_0-1}, A_{T_0-1}, R_{T_0}$
Episode 1	$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T_1-1}, A_{T_1-1}, R_{T_1}$
Episode 2	$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T_2-1}, A_{T_2-1}, R_{T_2}$
Episode 3	$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T_3-1}, A_{T_3-1}, R_{T_3}$

First-visit MC prediction algorithm

Identify the first time a state is visited and average the following returns

Monte Carlo Methods

First-visit MC prediction idea

v(**s**) = mean of

Episode s, a, r, **s**, a, r, **s**, a, r ,**s**, a, r, **s** ,a, r, s (terminal state)

v(**s**) = mean of

Monte Carlo Methods

First-visit MC prediction algorithm

Episode s, a, r, s, a, r, s, a, r ,s, a, r, s ,a, r, s (terminal state)

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo Methods

How to identify the optimal policy?

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy policy approach

$$\begin{aligned}\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],\end{aligned}$$

Monte Carlo Methods

How to identify the optimal policy?

We do not have a mode!

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy policy approach

$$
\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

Monte Carlo Methods

How to identify the optimal policy?

We need to use the Q-value!

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy policy approach

$$
\begin{aligned}
\pi'(s) &\doteq \boxed{\arg\max_a q_\pi(s,a)} \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

Monte Carlo Methods

How to identify the optimal policy?

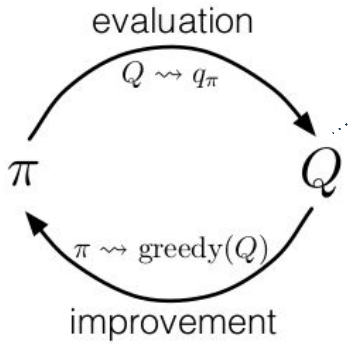
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy policy approach

$$
\begin{aligned}
\pi'(s) &\doteq \boxed{\arg\max_a q_\pi(s,a)} \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

We need to use the Q-value!

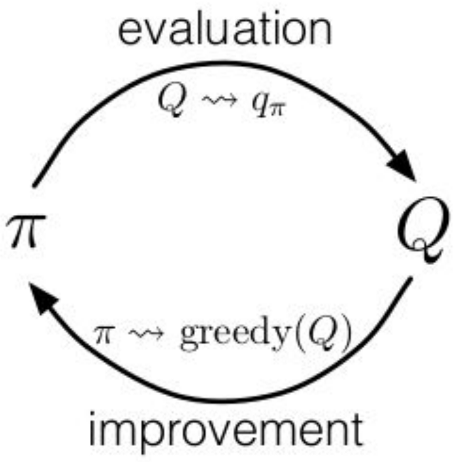
We need to estimate the action-value function!

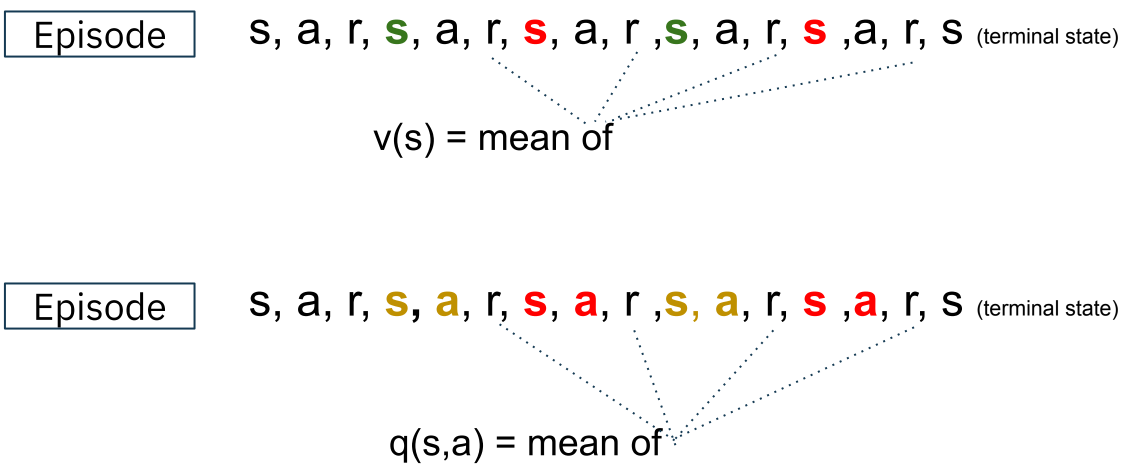


The same idea of policy iteration of DP but with Q + estimation

Monte Carlo Methods

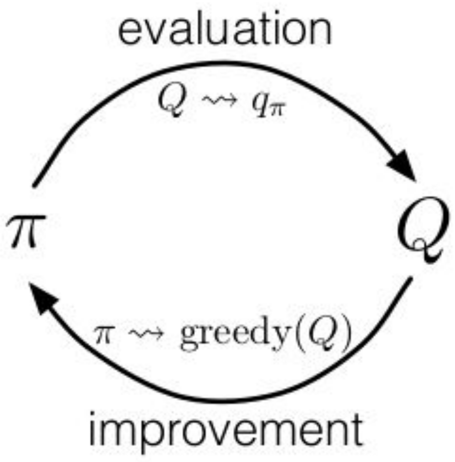
How to identify the optimal policy?





Monte Carlo Methods

How to identify the optimal policy?



Episode s, a, r, s, a, r, s, a, r ,s, a, r, s ,a, r, s (terminal state)

v(s) = mean of

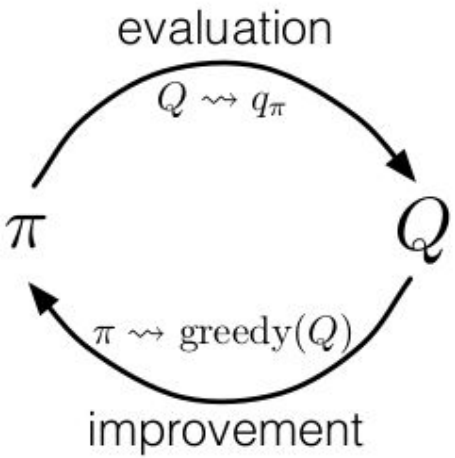
Episode s, a, r, s, a, r, s, a, r ,s, a, r, s ,a, r, s (terminal state)

q(s,a) = mean of

A policy might not generate all the pairs!
How can we guarantee exploration?

Monte Carlo Methods

How to identify the optimal policy?



Episode: s, a, r, s, a, r, s, a, r ,s, a, r, s ,a, r, s (terminal state)

v(s) = mean of

Episode: s, a, r, s, a, r, s, a, r ,s, a, r, s ,a, r, s (terminal state)

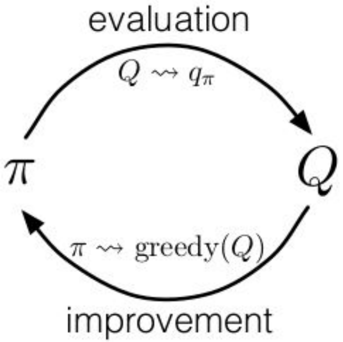
q(s,a) = mean of

A policy might not generate all the pairs!
How can we guarantee exploration?

Exploring start approach or ε-soft policy

Monte Carlo Methods

Control: how to find the optimal policy?



Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$

Exploring start approach

Evaluation

Improvement

Monte Carlo Methods

Control: how to find the optimal policy?

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

- $\pi \leftarrow$ an arbitrary ε-soft policy
- $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \text{argmax}_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

suboptimal

ε-soft policy

Evaluation

Improvement

Monte Carlo Methods

On-policy vs off-policy algorithms

Learning control methods dilemma

learning action-value of an optimal policy means also exploring …

Monte Carlo Methods

On-policy vs off-policy algorithms

Learning control methods dilemma
learning action-value of an optimal policy means also exploring …

On-policy algorithm
learning action-value for suboptimal policy

SARSA

Off-policy algorithm
Use two policies. One for exploring and one for searching the optimal policy

Q-learning

Temporal-Difference Learning

Combining Monte Carlo and Dynamic Programming

Temporal Difference Learning

Combining two ideas for prediction

Wait until the end of the episode

Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[G_t - V(S_t) \Big]$$

Wait until the next step

Temporal Difference Learning

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$

One step sampling

Estimating (as DP)

Temporal Difference Learning

TD(0) for prediction

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Temporal Difference Learning

SARSA: on-policy TD control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

On-policy update

Q is updated using the action A' derived from the actual policy

Temporal Difference Learning

Q-learning: off-policy TD control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

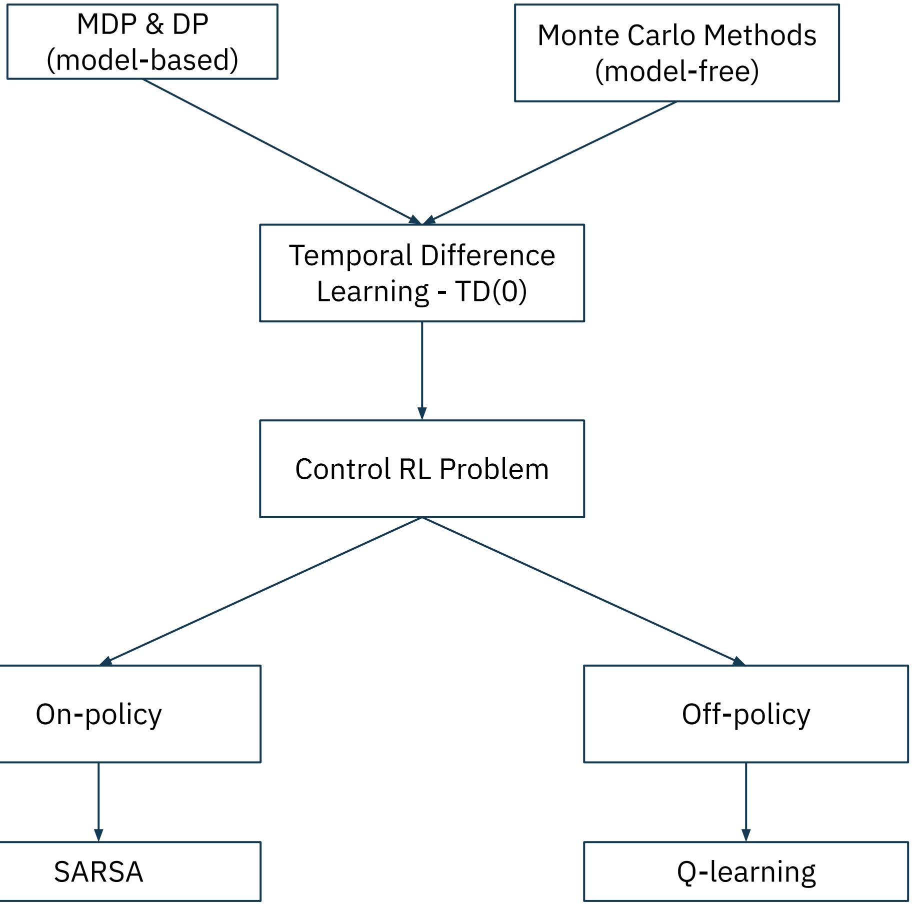
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$
    until S is terminal

Off-policy update

Q is updated using the greedy action a

Recap


MDP & DP
(model-based)
Monte Carlo Methods
(model-free)
Temporal Difference
Learning - TD(0)
Control RL Problem
On-policy
Off-policy
SARSA
Q-learning

Thank you