REINFORCEMENT LEARNING
READING GROUP

01

SESSION

Theme · Value-based methods

Reading: Murphy, Ch. 1 & §§2.1–2.4

rl-readinggroup.github.io

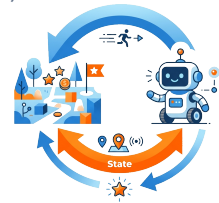
Introduction to Reinforcement Learning

From MDPs and dynamic programming to learning from experience

Mario Chacón-Falcón

Institute of Mathematical Sciences (ICMAT-CSIC)

November 10, 2026



Outline

1 Motivation

2 Formalization

3 Bellman equations & dynamic programming

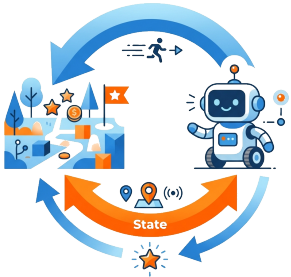
4 Learning from experience

5 Conclusions

Welcome

- RL and sequential decision making
- Main text: Murphy, *Reinforcement Learning: An Overview*
- Every two weeks, 12:00 (Madrid time), ∼45 min + discussion
- Recorded; slides and videos on the website
- Let us know if you want to present!

Web	rl-readinggroup.github.io
Discord	discord.gg/uakBBee7Uv
Mailing list	`rl_reading_group` (listas.csic.es)
GitHub	RL-readingGroup



The roadmap: 12 sessions, 6 themes



Outline

1 Motivation

2 Formalization

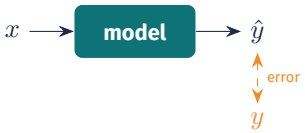
3 Bellman equations & dynamic programming

4 Learning from experience

5 Conclusions

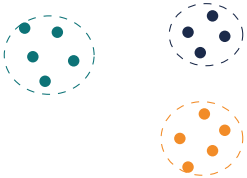
RL within machine learning

Supervised



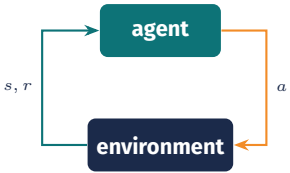
learn from labels

Unsupervised



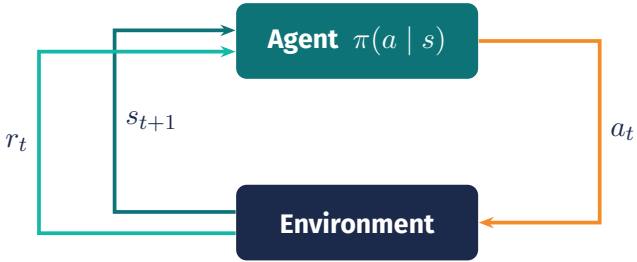
find structure

Reinforcement



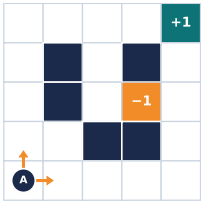
learn from interaction

Reinforcement Learning

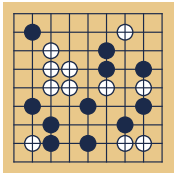


$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$$

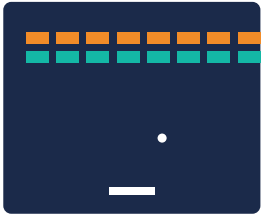
Classic examples



Mazes & navigation

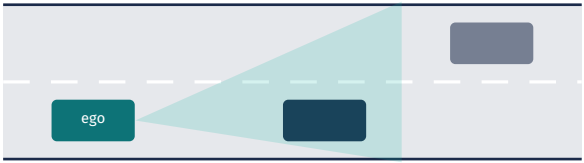


Go: AlphaGo (2016), AlphaZero (2018)

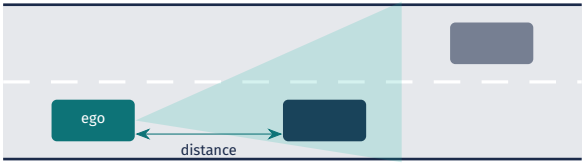


Atari: DQN (2015)

Example: a self-driving car

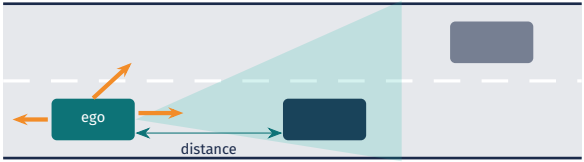


Example: a self-driving car



State: position, speed, distances to cars

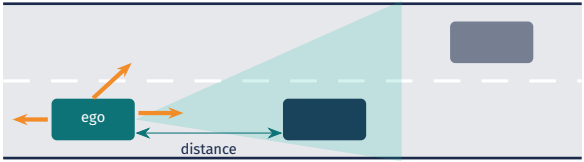
Example: a self-driving car



State: position, speed, distances to cars

Actions: steer, accelerate, brake

Example: a self-driving car

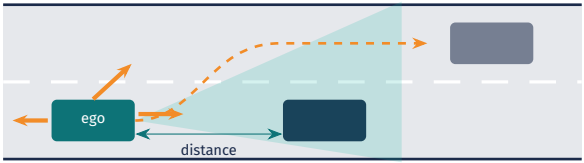


State: position, speed, distances to cars

Actions: steer, accelerate, brake

Reward: + progress, − crash, − harsh manoeuvres

Example: a self-driving car

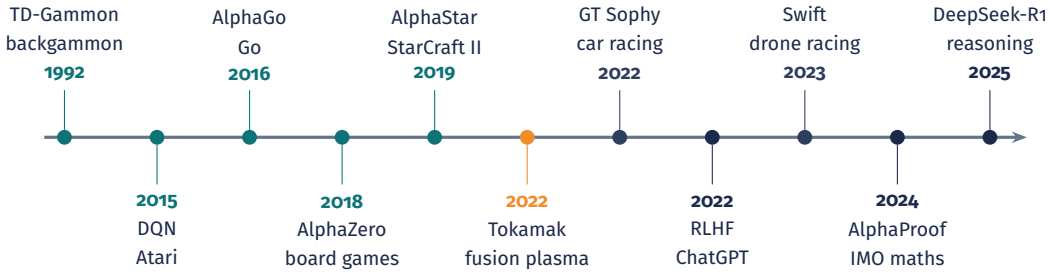


State: position, speed, distances to cars

Actions: steer, accelerate, brake

Reward: + progress, − crash, − harsh manoeuvres

RL in the real world



Outline

1 Motivation

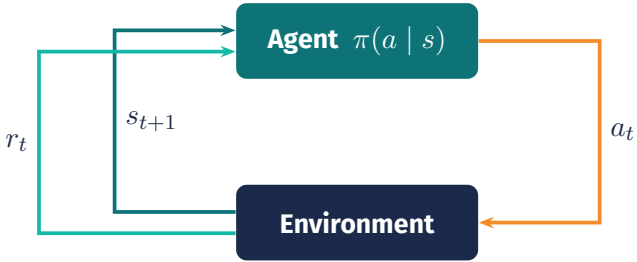
2 Formalization

3 Bellman equations & dynamic programming

4 Learning from experience

5 Conclusions

The environment interaction



$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$$

Markov decision processes

A **Markov decision process** is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p_S, p_R, p_0, \gamma)$:

$\mathcal{S}$	state space	
$\mathcal{A}$	action space	
$p_S(s' \mid s, a)$	transition function	$= \mathbb{P}(s_{t+1} = s' \mid s_t = s,\, a_t = a)$
$p_R(r \mid s, a, s')$	reward distribution	
$R(s, a)$	expected reward	$= \mathbb{E}[r_t \mid s_t = s,\, a_t = a]$
$p_0(s)$	initial state distribution	$= \mathbb{P}(s_0 = s)$

Markov property: $\mathbb{P}(s_{t+1} \mid s_{0:t}, a_{0:t}) = p_S(s_{t+1} \mid s_t, a_t)$

Policies, trajectories and the objective

Policy: a rule to choose actions

$$\pi(a \mid s) = \mathbb{P}(a_t = a \mid s_t = s) \qquad \text{(deterministic: } a = \pi(s)\text{)}$$

Trajectory: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, with

$$p(\tau) = p_0(s_0) \prod_{t \geq 0} \pi(a_t \mid s_t)\, p_S(s_{t+1} \mid s_t, a_t)\, p_R(r_t \mid s_t, a_t, s_{t+1})$$

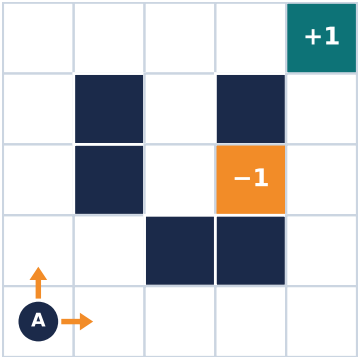
Return: total reward from time t until the episode ends at T

$$G_t = r_t + r_{t+1} + \cdots + r_{T-1} = \sum_{k=0}^{T-t-1} r_{t+k}$$

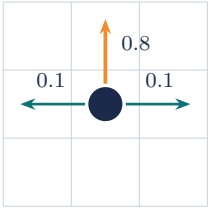
Objective

$$\pi^* = \arg\max_{\pi} J(\pi), \qquad J(\pi) = \mathbb{E}_{\pi}\left[G_0\right]$$

Running example: a stochastic maze



- $\mathcal{A} = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$
- $+1$ exit, -1 trap
- -0.04 per step



slippery floor

Finite vs infinite horizon

Episodic

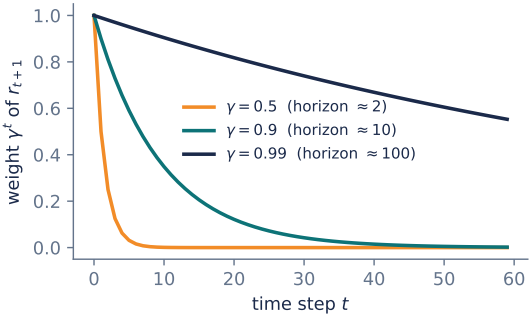


$$G_t = \sum_{k=0}^{T-t-1} r_{t+k}$$

Continuing



$$G_t = \sum_{k\geq 0} \gamma^k r_{t+k}, \quad \gamma < 1$$



horizon $\approx \dfrac{1}{1-\gamma}$ $\qquad |G_t| \leq \dfrac{r_{\max}}{1-\gamma}$

Outline

1 Motivation

2 Formalization

3 Bellman equations & dynamic programming

4 Learning from experience

5 Conclusions

Value functions and the Bellman equation

Objective

$$\pi^* = \arg\max_{\pi} J(\pi), \qquad J(\pi) = \mathbb{E}_{\pi}\left[G_0\right]$$

Value functions and the Bellman equation

Objective

$$\pi^* = \arg\max_{\pi} J(\pi), \qquad J(\pi) = \mathbb{E}_{\pi}\left[G_0\right]$$

State value

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s\right]$$

Action value

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s, a_t = a\right]$$

Value functions and the Bellman equation

Objective

$$\pi^* = \arg\max_{\pi} J(\pi), \qquad J(\pi) = \mathbb{E}_\pi\big[G_0\big]$$

State value

$$V^\pi(s) = \mathbb{E}_\pi\big[G_t \mid s_t = s\big]$$

Action value

$$Q^\pi(s,a) = \mathbb{E}_\pi\big[G_t \mid s_t = s, a_t = a\big]$$

Bellman equations:

$$V^\pi(s) = \sum_a \pi(a \mid s)\Big[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V^\pi(s')\Big]$$

Policy evaluation

How good is a given policy π? Turn the Bellman equation into an **update rule**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a \mid s)\Big[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V_k(s')\Big] \qquad \forall s \in \mathcal{S}$$

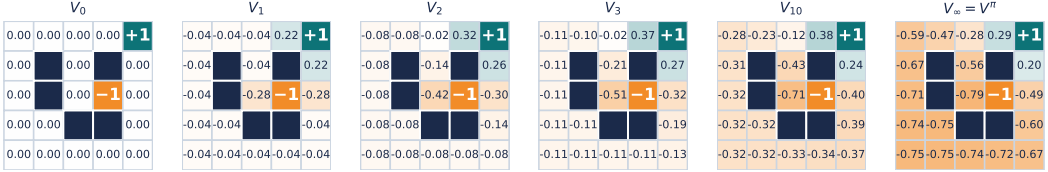
$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V^\pi$$

Policy evaluation

How good is a given policy π? Turn the Bellman equation into an **update rule**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a \mid s)\Big[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V_k(s')\Big] \qquad \forall s \in \mathcal{S}$$

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V^\pi$$



maze, π uniformly random, starting from $V_0 = 0$

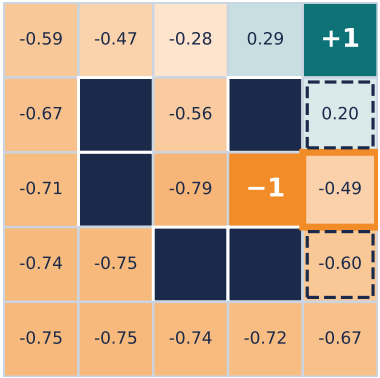
Policy improvement

V^{π}, π random

-0.59	-0.47	-0.28	0.29	**+1**
-0.67		-0.56		0.20
-0.71		-0.79	**−1**	-0.49
-0.74	-0.75			-0.60
-0.75	-0.75	-0.74	-0.72	-0.67

Policy improvement

V^π, π random



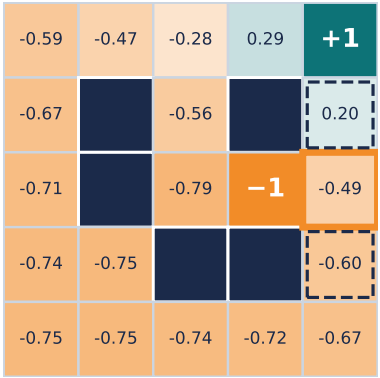
Take a now, then follow π:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V^\pi(s')$$

Better than π if $Q^\pi(s,a) > V^\pi(s)$

Policy improvement

V^π, π random



cell right of the trap, π uniformly random

a	$Q^\pi(s,a)$
$\uparrow$	$\mathbf{-0.03}$
$\rightarrow$	-0.45
$\downarrow$	-0.64
$\leftarrow$	-0.85

Evaluation: $V^\pi(s) = \frac{1}{4}\sum_a Q^\pi(s,a) = -0.49$

Improvement: $\pi'(s) = \arg\max_a Q^\pi(s,a) = \uparrow$

$$Q^\pi(s,\uparrow) = -0.03 \;\geq\; V^\pi(s) = -0.49$$

Policy improvement

Theorem (Policy improvement). *If $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$ for all s, then $V^{\pi'}(s) \geq V^{\pi}(s)$ for all s.*

If nothing improves, $V^{\pi'} = V^{\pi}$ and we've reached an optimal policy.

Greedy policy: act well for one step, then follow π

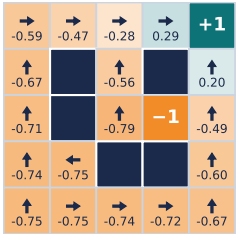
$$\pi'(s) = \arg\max_{a} Q^{\pi}(s, a)$$

Policy iteration

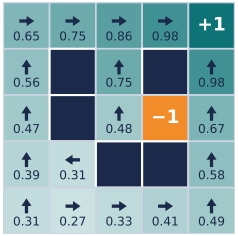
$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

E: policy evaluation **I**: policy improvement (greedy)

V^{π_0} and π_1

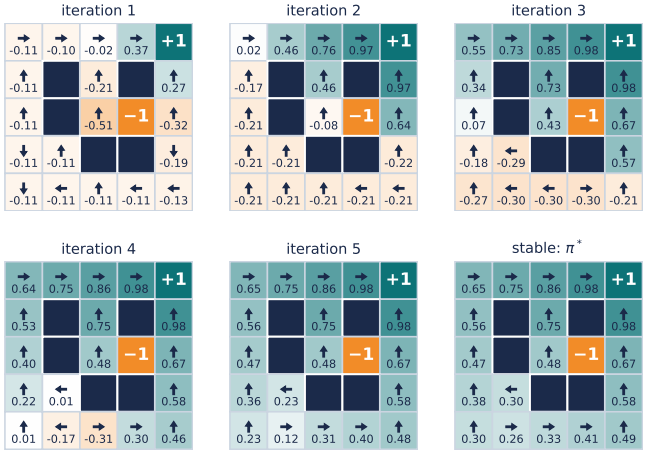


$V^{\pi_1} = V^*$ and $\pi_2 = \pi_1$



- $V^{\pi_0} \leq V^{\pi_1} \leq V^{\pi_2} \leq \cdots \leq V^*$

Value iteration



each iteration:

- 3 sweeps of policy evaluation (values)
- greedy improvement (arrows)

The catch

- DP needs the **model**: $p_S(s' \mid s, a)$ and $R(s, a)$
- DP is **expensive**: Go has $\sim 10^{170}$ states

What if we don't know the model?

Outline

1 Motivation

2 Formalization

3 Bellman equations & dynamic programming

4 Learning from experience

5 Conclusions

Monte Carlo prediction

$$V^{\pi}(s) = \underbrace{\mathbb{E}_{\pi}\left[G_t \mid s_t = s \right]}_{\text{Monte Carlo: sample it}} = \underbrace{\mathbb{E}_{\pi}\left[r_t + \gamma V^{\pi}(s_{t+1}) \mid s_t = s \right]}_{\text{DP: compute it with the model}}$$

Monte Carlo prediction

$$V^\pi(s) \;=\; \underbrace{\mathbb{E}_\pi\big[\, G_t \mid s_t = s \,\big]}_{\text{Monte Carlo: sample it}} \;=\; \underbrace{\mathbb{E}_\pi\big[\, r_t + \gamma\, V^\pi(s_{t+1}) \mid s_t = s \,\big]}_{\text{DP: compute it with the model}}$$

Run episodes with π and **average the returns** observed from each state:

$$V(s_t) \;\leftarrow\; V(s_t) + \alpha\,\big[\, G_t - V(s_t) \,\big]$$

Monte Carlo prediction

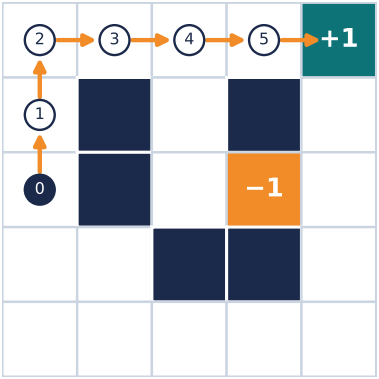
$$V^\pi(s) \;=\; \underbrace{\mathbb{E}_\pi\big[\, G_t \mid s_t = s \,\big]}_{\text{Monte Carlo: sample it}} \;=\; \underbrace{\mathbb{E}_\pi\big[\, r_t + \gamma\, V^\pi(s_{t+1}) \mid s_t = s \,\big]}_{\text{DP: compute it with the model}}$$

Run episodes with π and **average the returns** observed from each state:

$$V(s_t) \;\leftarrow\; V(s_t) + \alpha\,\big[\, G_t - V(s_t) \,\big]$$

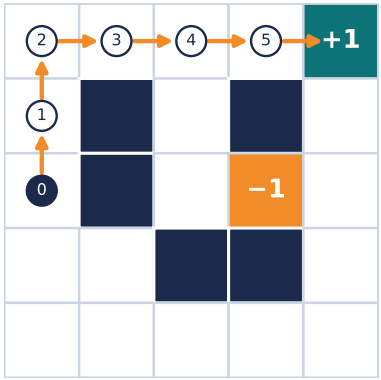
- $\alpha = 1/N(s_t)$
- Only sampled episodes
- we must wait until the episode ends to know G_t

Monte Carlo: Example



one episode from s_0 following a policy π

Monte Carlo: Example

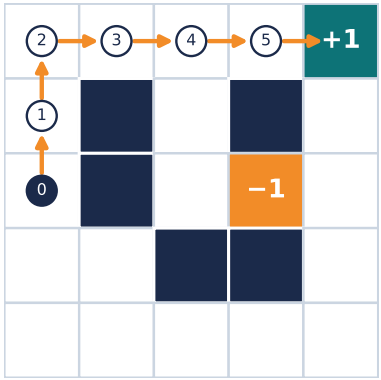


one episode from s_0 following a policy π

Returns, computed backwards: $G_t = r_t + \gamma\, G_{t+1}$

t	a_t	r_t	G_t
5	$\rightarrow$	$+1$	1.00
4	$\rightarrow$	-0.04	0.91
3	$\rightarrow$	-0.04	0.82
2	$\rightarrow$	-0.04	0.74
1	$\uparrow$	-0.04	0.67
0	$\uparrow$	-0.04	$\mathbf{0.59}$

Monte Carlo: Example



one episode from s_0 following a policy π

Returns, computed backwards: $G_t = r_t + \gamma\, G_{t+1}$

t	a_t	r_t	G_t
5	$\rightarrow$	$+1$	1.00
4	$\rightarrow$	-0.04	0.91
3	$\rightarrow$	-0.04	0.82
2	$\rightarrow$	-0.04	0.74
1	$\uparrow$	-0.04	0.67
0	$\uparrow$	-0.04	$\mathbf{0.59}$

$$G_0 = 0.59$$

Temporal-difference learning, TD(0)

Replace G_t by one step plus our current guess

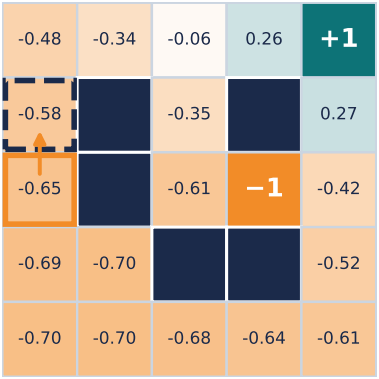
$$V(s_t) \leftarrow V(s_t) + \alpha \left[\underbrace{r_t + \gamma V(s_{t+1})}_{\text{TD target}} - V(s_t) \right]$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \qquad \text{(TD error)}$$

- samples like Monte Carlo, **bootstraps** like DP
- online, works for continuing tasks

TD: Example

V (TD, 50 episodes)

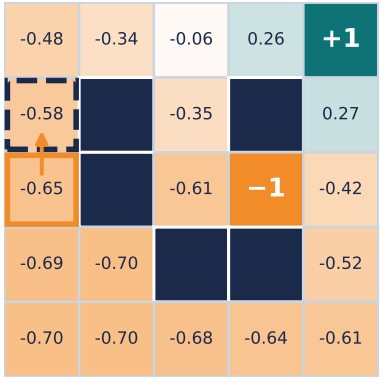


first step of the same episode: $s_0 \xrightarrow{\uparrow} s_1$,

$$r_0 = -0.04$$

TD: Example

V (TD, 50 episodes)



first step of the same episode: $s_0 \xrightarrow{\uparrow} s_1$,

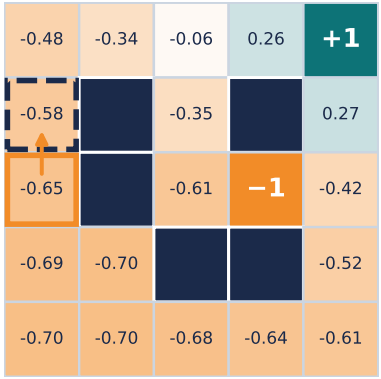
$$r_0 = -0.04$$

$$V(s_0) = -0.649, \qquad V(s_1) = -0.578$$

$$\delta_0 = -0.04 + 0.95 \cdot (-0.578) - (-0.649) = 0.060$$

TD: Example

V (TD, 50 episodes)



first step of the same episode: $s_0 \xrightarrow{\uparrow} s_1$, $r_0 = -0.04$

$$V(s_0) = -0.649, \qquad V(s_1) = -0.578$$

$$\delta_0 = -0.04 + 0.95 \cdot (-0.578) - (-0.649) = 0.060$$

$$V(s_0) \leftarrow -0.649 + 0.1 \cdot 0.060 = -0.643$$

$\alpha = 0.1$; values from 50 episodes of TD

Monte Carlo and TD comparison

MC, 10 episodes

-0.61	-0.58	-0.37	0.21	+1
-0.53		-0.13		0.53
-0.69		-1.00	−1	-0.75
-0.74	-0.75			-0.71
-0.74	-0.74	-0.76	-0.75	-0.78

MC, 100 episodes

-0.59	-0.48	-0.35	0.39	+1
-0.64		-0.45		0.44
-0.72		-0.81	−1	-0.36
-0.75	-0.76			-0.55
-0.75	-0.75	-0.73	-0.72	-0.68

MC, 1000 episodes

-0.60	-0.52	-0.33	0.25	+1
-0.68		-0.55		0.26
-0.72		-0.79	−1	-0.48
-0.74	-0.75			-0.61
-0.75	-0.75	-0.74	-0.72	-0.68

DP: V^{π}

-0.59	-0.47	-0.28	0.29	+1
-0.67		-0.56		0.20
-0.71		-0.79	−1	-0.49
-0.74	-0.75			-0.60
-0.75	-0.75	-0.74	-0.72	-0.67

TD, 10 episodes

-0.17	-0.16	-0.03	0.20	+1
-0.21		-0.04		0.04
-0.23		-0.19	−1	-0.32
-0.25	-0.23			-0.12
-0.25	-0.22	-0.19	-0.18	-0.16

TD, 100 episodes

-0.57	-0.44	-0.16	0.52	+1
-0.64		-0.47		0.23
-0.70		-0.73	−1	-0.24
-0.73	-0.74			-0.44
-0.74	-0.74	-0.73	-0.68	-0.61

TD, 1000 episodes

-0.63	-0.51	-0.41	0.04	+1
-0.67		-0.64		0.12
-0.69		-0.90	−1	-0.49
-0.75	-0.77			-0.67
-0.77	-0.77	-0.77	-0.75	-0.72

DP: V^{π}

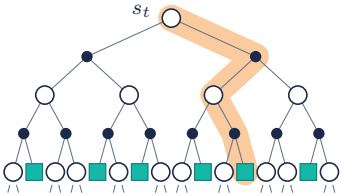
-0.59	-0.47	-0.28	0.29	+1
-0.67		-0.56		0.20
-0.71		-0.79	−1	-0.49
-0.74	-0.75			-0.60
-0.75	-0.75	-0.74	-0.72	-0.67

π random; both approach the DP solution without using the model

Comparing the backups

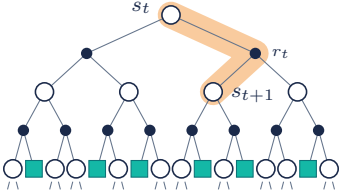
Monte Carlo

target: G_t



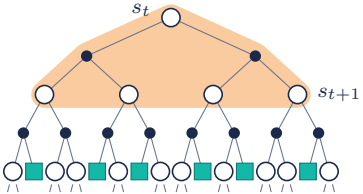
Temporal difference

target: $r_t + \gamma V(s_{t+1})$



Dynamic programming

target: $\mathbb{E}_\pi\left[r_t + \gamma V(s_{t+1})\right]$



state · action ■ terminal

Policy improvement with V:

$$\pi'(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V(s') \right]$$

SARSA

Policy improvement with V:

$$\pi'(s) = \arg\max_a \Big[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V(s') \Big]$$

$\Rightarrow$ learn $Q(s,a)$ instead: $\pi'(s) = \arg\max_a Q(s,a)$, no model needed

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \big[r_t + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t,a_t) \big]$$

SARSA

Policy improvement with V:

$$\pi'(s) = \arg\max_a \Big[R(s,a) + \gamma \sum_{s'} p_S(s' \mid s,a)\, V(s') \Big]$$

$\Rightarrow$ learn $Q(s,a)$ instead: $\pi'(s) = \arg\max_a Q(s,a)$, no model needed

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \big[r_t + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t,a_t) \big]$$

Caveat: we need to **explore**, or some actions are never tried

$$a_t = \begin{cases} \arg\max_a Q(s_t,a) & \text{w.p. } 1-\varepsilon \\ \text{random} & \text{w.p. } \varepsilon \end{cases} \qquad (\varepsilon\text{-greedy})$$

SARSA

Algorithm 1 SARSA

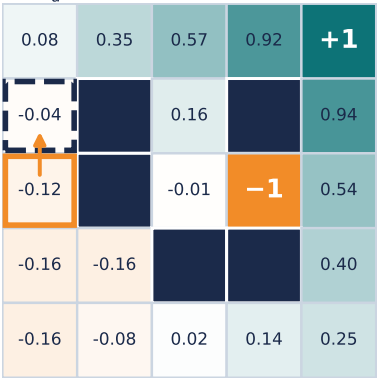
$Q(s, a) \leftarrow 0$
for each episode **do**
  $s \leftarrow s_0$; pick a ε-greedy from $Q(s, \cdot)$
  while s not terminal **do**
    take a, observe r, s'
    pick a' ε-greedy from $Q(s', \cdot)$
    $Q(s, a) \mathrel{+}= \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$, $a \leftarrow a'$
  end while
end for

$$(s_t, a_t, r_t, s_{t+1}, a_{t+1})$$

a_{t+1} from the **same** policy

SARSA: Example

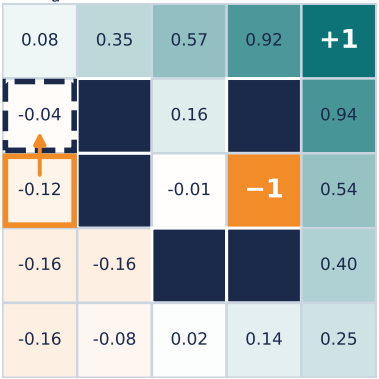
$\max_a Q(s, a)$ (SARSA, 100 ep.)



$$(s, a, r, s', a') = (s_0, \uparrow, -0.04, s_1, \uparrow)$$

SARSA: Example

$\max_a Q(s,a)$ (SARSA, 100 ep.)



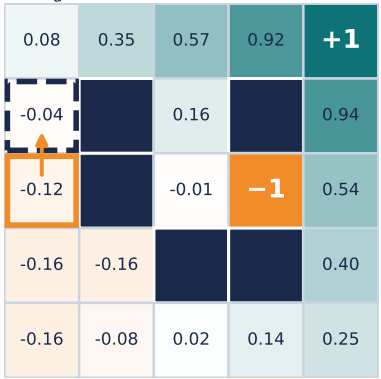
$$(s, a, r, s', a') = (s_0, \uparrow, -0.04, s_1, \uparrow)$$

$$Q(s_0, \uparrow) = -0.121, \qquad Q(s_1, \uparrow) = -0.038$$

target: $-0.04 + 0.95 \cdot (-0.038) = -0.076$

SARSA: Example

$\max_a Q(s,a)$ (SARSA, 100 ep.)



$$(s, a, r, s', a') = (s_0, \uparrow, -0.04, s_1, \uparrow)$$

$$Q(s_0, \uparrow) = -0.121, \qquad Q(s_1, \uparrow) = -0.038$$

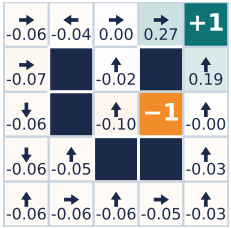
target: $-0.04 + 0.95 \cdot (-0.038) = -0.076$

$$Q(s_0, \uparrow) \leftarrow -0.121 + 0.1\,(-0.076 + 0.121) = -0.117$$

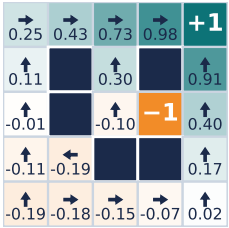
$\alpha = 0.1$, $\varepsilon = 0.1$; values from 100 episodes of SARSA

SARSA Example

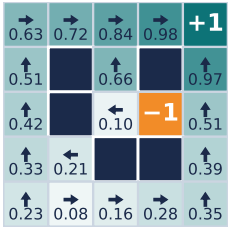
after 10 episodes



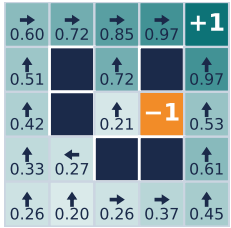
after 100 episodes



after 1000 episodes

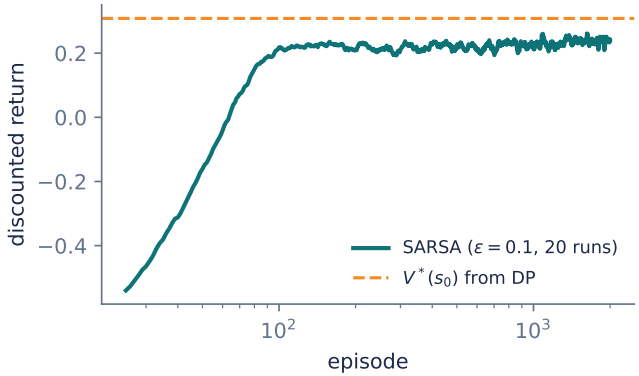


after 10000 episodes



colour: $\max_a Q(s, a)$ arrows: greedy action grey: not visited

SARSA learning curve



Outline

1 Motivation

2 Formalization

3 Bellman equations & dynamic programming

4 Learning from experience

5 Conclusions

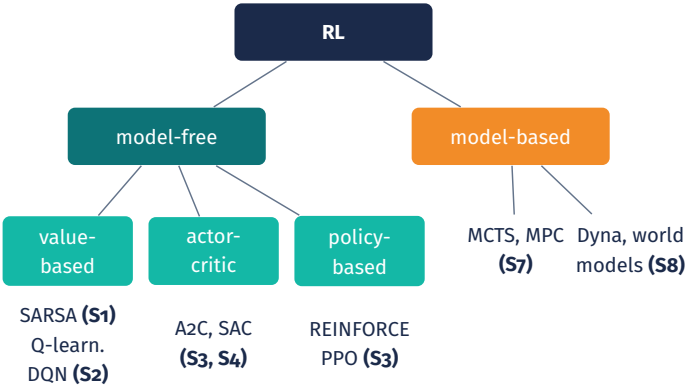
Conclusions

- Motivation: RL is sequential decision making learned from interaction, guided by a reward
- Formalization: MDPs, policies, returns and the objective
- Value functions and the Bellman equation
- With a model: dynamic programming (evaluation, improvement, iteration)
- Without a model: Monte Carlo, TD and SARSA learn from samples

Next steps

Today	Next	Session
a table with one value per state	function approximation, deep RL	2
act greedily on Q	optimise the policy directly	3–4
a given reward	rewards from preferences, LLMs	5
one objective	several objectives	6
model known (DP) or ignored (TD)	learn a model and plan	7–8
one agent	several agents, adversaries	9, 12
ε-greedy exploration	principled exploration, Bayesian RL	10
the state is fully observed	partial observability	11

A map of RL algorithms



References

- K. P. Murphy (2024). *Reinforcement Learning: An Overview*. arXiv:2412.05265. arxiv.org/abs/2412.05265
- R. S. Sutton and A. G. Barto (2018). *Reinforcement Learning: An Introduction*, 2nd edition. MIT Press.

Next session

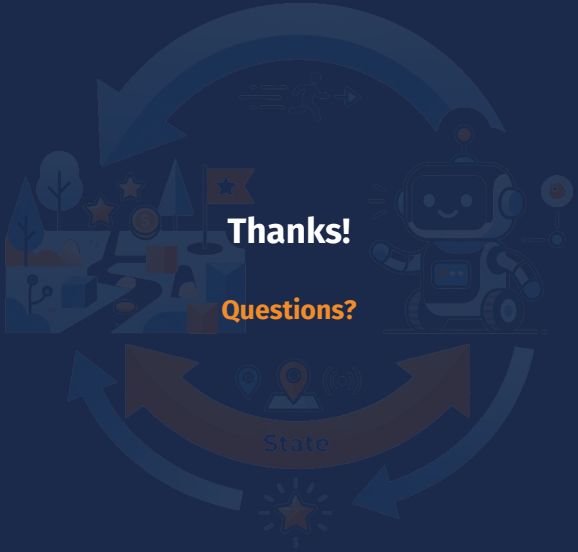
Deep Q-Learning & extensions

Murphy, §2.5

Daniel Corrales (ICMAT-CSIC)

November 25, 2026

rl-readinggroup.github.io/schedule · Discord



Thanks!

Questions?