

Introduction to Reinforcement Learning

https://github.com/racousin/rl_introduction

Raphael Cousin


March 4, 2024

Course Objectives

- The keys to go by yourself in RL
- Practice coding
- General culture

What do you already know about RL?

Why reinforcement learning?



alphago.jpg

Figure 1: In 2017, AlphaGo defeated Ke Jie, the world's top-ranked Go player.

RL vs. Other Machine Learning Paradigms

- Supervised Learning: Learning from labeled data to predict outcomes.
- Unsupervised Learning: Finding patterns in data without explicit labels.
- Reinforcement Learning: Learning decision-making by interacting with an environment to achieve goals.

In contrast to the other two, RL is focused on learning from the consequences of actions.

The RL Framework

- Absence of explicit "correct" actions.
- Learning is guided by rewards that represent the objectives.

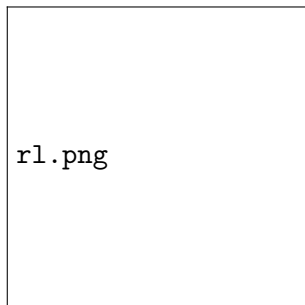


Figure 2: The agent-environment interaction in RL.

Applications of RL

RL has been successfully applied in various fields, including:

- Autonomous vehicles/Robotics
- Control systems
- Chat bots policy
- Marketing/Trading strategies
- Game playing and beyond

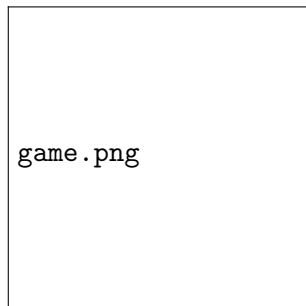


Figure 3: RL in video games.

Course Outline

- I) Understanding Markov Decision Processes (MDPs)
- II) Exploring Model-Free Reinforcement Learning
- III) Diving into Deep Reinforcement Learning

We'll start with the basics and gradually move to more complex concepts.

I) Understanding Markov Decision Processes

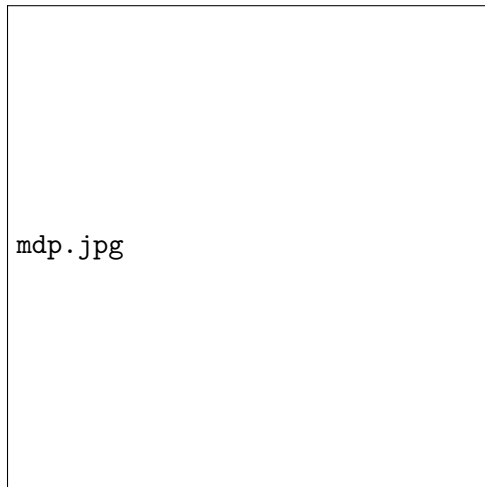


Figure 4: Illustrative example of an MDP, showcasing state transitions, actions, and rewards.

First Glossary of MDPs

- State Space (S)
- Action Space (A)
- Transition Model (P)
- Reward function (R)
- Policy (π)
- Trajectory (τ)
- Return (G)

Simple Grid World Problem

Our environment is a 4x4 grid where an agent aims to reach a goal.

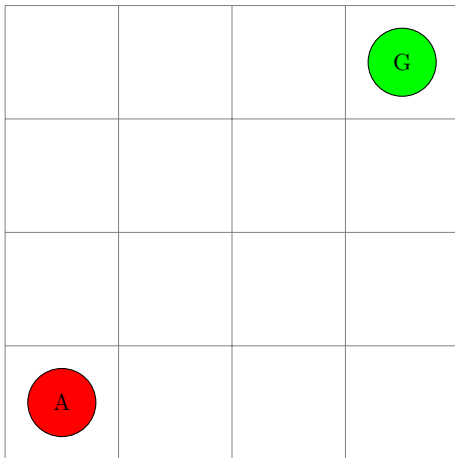


Figure 5: A: Agent, G: Goal

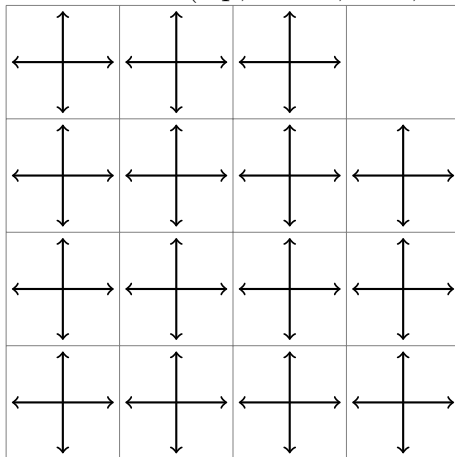
State Space (S)

16 discrete states.

$S_{0,3}$	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$
$S_{0,2}$	$S_{1,2}$	$S_{2,2}$	$S_{3,2}$
$S_{0,1}$	$S_{1,1}$	$S_{2,1}$	$S_{3,1}$
$S_{0,0}$	$S_{1,0}$	$S_{2,0}$	$S_{3,0}$

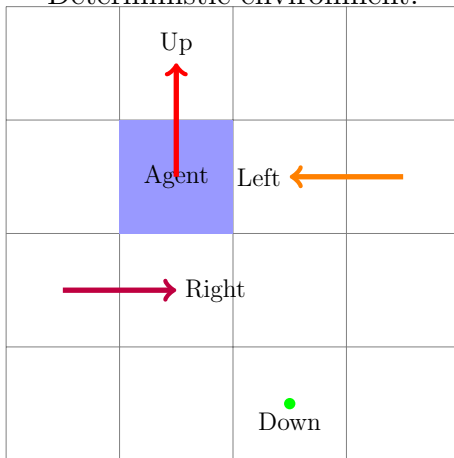
Action Space (A)

4 discrete actions (Up, Down, Left, Right).



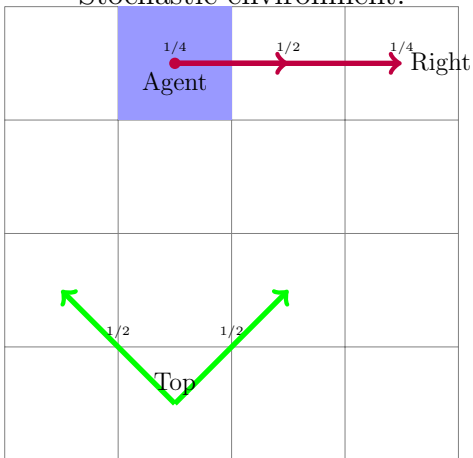
Transition Model: $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$

Deterministic environment.



Transition Model: $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$

Stochastic environment.



Reward function: $r = R(s, a) = r(s')$

Simple goal reward.

0	0	0	+1
0	0	0	0
0	0	0	0
0	0	0	0

Reward function: $r = R(s, a) = r(s')$

Other example of environment reward function.

-43	-64	-51	+1
-39	-28	-67	-2
-21	-30	-82	-20
-13	-83	-7	-63

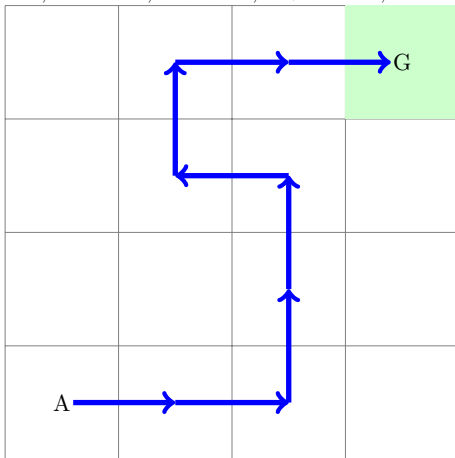
Policy: ($\pi : S \rightarrow A$)

Agent action in a state defined by its policy
deterministic/stochastic

<div>0.31 ↕ 0.27 ← → 0.30 ↕ 0.11</div>	<div>0.13 ↕ 0.03 ← → 0.35 ↕ 0.49</div>	<div>0.18 ↕ 0.14 ← → 0.25 ↕ 0.43</div>	
<div>0.03 ↕ 0.29 ← → 0.46 ↕ 0.22</div>	<div>0.19 ↕ 0.03 ← → 0.51 ↕ 0.28</div>	<div>0.29 ↕ 0.38 ← → 0.06 ↕ 0.27</div>	<div>0.25 ↕ 0.29 ← → 0.26 ↕ 0.21</div>
<div>0.08 ↕ 0.46 ← → 0.26 ↕ 0.20</div>	<div>0.31 ↕ 0.22 ← → 0.31 ↕ 0.16</div>	<div>0.31 ↕ 0.15 ← → 0.50 ↕ 0.04</div>	<div>0.27 ↕ 0.22 ← → 0.20 ↕ 0.31</div>
<div>0.16 ↕ 0.34 ← → 0.33 ↕ 0.17</div>	<div>0.12 ↕ 0.16 ← → 0.04 ↕ 0.68</div>	<div>0.38 ↕ 0.01 ← → 0.18 ↕ 0.44</div>	<div>0.47 ↕ 0.08 ← → 0.24 ↕ 0.21</div>

Trajectory: $\tau_\pi = (s_0, a_0, s_1, a_1, \dots)$

$(s_{0,0}, \rightarrow, 0, s_{1,0}, \rightarrow, 0, s_{2,0}, \uparrow, 0, s_{2,1}, \uparrow, 0, s_{2,2}, \leftarrow, 0, s_{1,2}, \uparrow, 0, s_{1,3}, \rightarrow, 0, s_{2,3}, \rightarrow, 1)$



Return: $G_t = \sum_{k=1}^T \gamma^k r_{t+k}$

Cumulative rewards

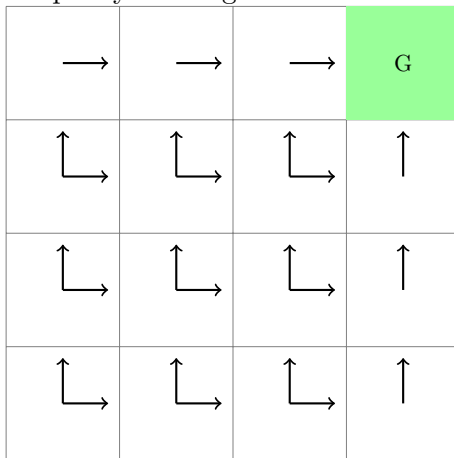
	t=6 1	t=7 1	G
	t=5 1	t=4 1	
		t=3 1	
t=0 1	t=1 1	t=2 1	

Discounted rewards (0.95)

	t=6 0.90	t=7 0.95	G
	t=5 0.86	t=4 0.81	
		t=3 0.77	
t=0 0.66	t=1 0.7	t=2 0.74	

Objective: Find best Policy $\pi^* = \arg \max_{\pi} E_{\tau \sim \pi}[G(\tau)]$

Optimal policy in the grid world environment.



Let's Code: Environment and Agent Interaction

Second Glossary of MDPs

- Value Function (V)
- Action Value Function (Q)
- Bellman Equations
- Dynamic Programming

Value Function: $V^\pi(s) = E_{\tau \sim \pi}[G_t | S_t = s]$

Expected Return for State following π

0.61	0.78	0.94	G
0.44	0.61	0.78	0.94
0.28	0.44	0.61	0.78
0.11	0.28	0.44	0.61

Action Value Function:

$$Q^\pi(s, a) = E_{\tau \sim \pi}[G_t | S_t = s, A_t = a]$$

Expected Return for State-Action following π

<div> <div>0.50</div> <div>0.50</div> <div>0.67</div> <div>0.33</div> </div>	<div> <div>0.67</div> <div>0.50</div> <div>0.83</div> <div>0.50</div> </div>	<div> <div>0.83</div> <div>0.67</div> <div>1.00</div> <div>0.67</div> </div>	G
<div> <div>0.50</div> <div>0.33</div> <div>0.50</div> <div>0.17</div> </div>	<div> <div>0.67</div> <div>0.33</div> <div>0.67</div> <div>0.33</div> </div>	<div> <div>0.83</div> <div>0.50</div> <div>0.83</div> <div>0.50</div> </div>	<div> <div>1.00</div> <div>0.67</div> <div>0.83</div> <div>0.67</div> </div>
<div> <div>0.33</div> <div>0.17</div> <div>0.33</div> <div>0.17</div> </div>	<div> <div>0.50</div> <div>0.17</div> <div>0.50</div> <div>0.17</div> </div>	<div> <div>0.67</div> <div>0.33</div> <div>0.67</div> <div>0.33</div> </div>	<div> <div>0.83</div> <div>0.50</div> <div>0.67</div> <div>0.50</div> </div>
<div> <div>0.17</div> <div>0.17</div> <div>0.17</div> <div>0.17</div> </div>	<div> <div>0.33</div> <div>0.17</div> <div>0.33</div> <div>0.17</div> </div>	<div> <div>0.50</div> <div>0.17</div> <div>0.50</div> <div>0.17</div> </div>	<div> <div>0.67</div> <div>0.33</div> <div>0.50</div> <div>0.50</div> </div>

Bellman Equations

- **Idea :** The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next.

$$V(s) = \mathbb{E}[G_t | S_t = s]$$

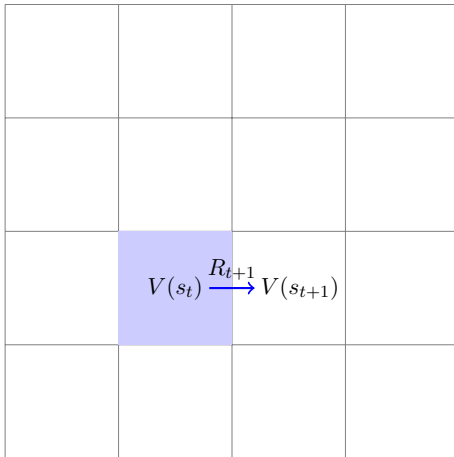
$$= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]$$

$$Q(s, a) = \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a]$$

Value Function Decomposition: $V^\pi(s)$

Value Function: $V^\pi(s) = E[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s]$



Bellman Equations development

$$V_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_{\pi}(s, a)$$

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_{\pi}(s')$$

$$V_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_{\pi}(s') \right)$$

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_{\pi}(s', a')$$

The MDP Solution

Dynamic Programming allows to resolve the MDP optimization problem ($\pi^* = \arg \max_{\pi} E_{\tau \sim \pi}[G(\tau)]$). It is an iterative process:

- Policy initialization
- Policy evaluation
- Policy improvement

Policy evaluation

Policy Evaluation: compute the state-value V_π for a given policy π :
We initialize V_0 arbitrarily. And we update it using:

$$\begin{aligned} V_{k+1}(s) &= \mathbb{E}_\pi[r + \gamma V_k(s_{t+1}) | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) (R(s, a) + \gamma V_k(s')) \quad (1) \end{aligned}$$

$V_\pi(s)$ is a fix point for (1), so if $(V_k)_{k \in \mathbb{N}}$ converges, it converges to V_π .

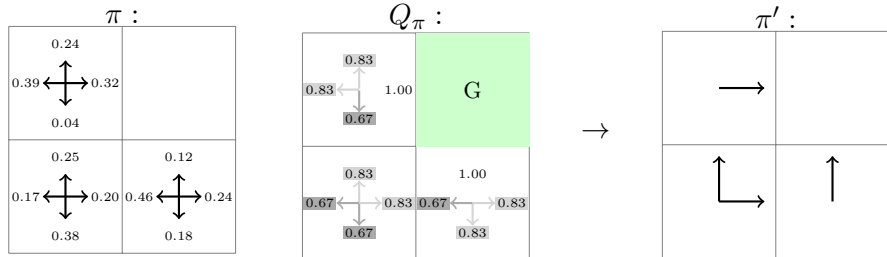
Policy Improvement

Policy Improvement: generates a better policy $\pi' \geq \pi$ by acting greedily. Compute Q from V ($\forall a, s$):

$$\begin{aligned} Q_{\pi}(s, a) &= \mathbb{E}[R_{t+1} + \gamma V_{\pi}(S_{t+1}) | S_t = s, A_t = a] \\ &= \sum_{s'} P(s' | s, a) (R(s, a) + \gamma V_{\pi}(s')) \end{aligned}$$

Update greedily: $\pi'(s) = \arg \max_{a \in \mathcal{A}} Q_{\pi}(s, a)$ ($\forall s$)

Policy improvement: $\pi'(s) = \arg \max_{a \in A} Q_{\pi}(s, a)$



Policy Iteration: iterative procedure to improve the policy when combining policy evaluation and improvement.

$$\pi_0 \xrightarrow{\text{evaluation}} V_{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{evaluation}} \dots \xrightarrow{\text{improve}} \pi_* \xrightarrow{\text{evaluation}} V_* \quad (1)$$

Bellman Equations Optimality

Bellman equations for the optimal value functions

$$V_*(s) = \max_{a \in \mathcal{A}} Q_*(s, a)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_*(s')$$

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_*(s') \right)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \max_{a' \in \mathcal{A}} Q_*(s', a')$$

Take home message

Initialize $\pi(s), \forall s$

- 1 Evaluate $V_\pi(s), \forall s$ (using $\mathbb{P}_{ss'}^a$)
- 2 Compute $Q_\pi(s, a), \forall s, a$ (using $\mathbb{P}_{ss'}^a$)
- 3 Update $\pi'(s) = \max_a Q_\pi(s, a), \forall s$
- 4 While $\pi'(s) \neq \pi(s)$ do $\pi(s) = \pi'(s)$ and iterate

Result : $\pi = \arg \max_\pi E[G]$

Coding session

Dynamic Programming