Faculté des **sciences économiques** et de **gestion**
Université de Strasbourg

Reinforcement Learning

UE 2 Machine learning

Nattirat Mayer

Fall-Winter Semester 2025-26

About Me: Academic & Professional Background

- **Bachelor of Engineering (Aerospace)**
  RMIT University, Australia (2017)
- **Industry Experience** R&D roles at:
  - KSB AG, Germany (Fluid Mechanic Systems)
  - Continental AG, Germany (Automotive Systems)
  - Accenture Inc, Thailand (Technology Consulting)
- **Master of Data Science in Economics (DS2E)**
  University of Strasbourg, France (2024)
- **PhD Candidate**
  Bureau of Theoretical and Applied Economics (BETA)
  University of Strasbourg (Since 2024)

Course details

- Contact
  - Email: npromwang@unistra.fr
  - Office hours: after the lecture and by appointment
- Lecture and tutorial: 20 hours on

Session	Room	Duration (h)	Start	End
Monday 15/09/2025	A 330 - Idem-Lab	02h00	14h00	16h00
Thursday 18/09/2025	A 330 - Idem-Lab	02h00	14h00	16h00
Thursday 25/09/2025	A 330 - Idem-Lab	02h00	13h00	15h00
Friday 26/09/2025	A 330 - Idem-Lab	02h00	09h00	11h00
Wednesday 15/10/2025	A 330 - Idem-Lab	02h00	13h00	15h00
Friday 17/10/2025	A 330 - Idem-Lab	02h00	09h00	11h00
Tuesday 21/10/2025	A 330 - Idem-Lab	02h00	10h00	12h00
Wednesday 22/10/2025	A 330 - Idem-Lab	02h00	10h00	12h00
Thursday 23/10/2025	A 330 - Idem-Lab	02h00	10h00	12h00
Friday 24/10/2025	A 330 - Idem-Lab	02h00	10h00	12h00

- Each session covers a lecture and a tutorial (hand calculations and coding exercises).

Course evaluation

At the level of the UE, the final grade for the written exam and the project (report + presentation) will be the average of different evaluations

- **Exam**: Coefficient = 0.4
  - Written exam: TBD. Duration: 30 min
  - The final written exam covers the material of the entire lecture.

Course evaluation (continue)

- **Project (report + presentation)**: Coefficient = 0.3+0.3
  Students (by groups of 2 students max. However, individual work is highly encouraged) work on one project, and prepare a dossier which will be evaluated and presented (15 min).
  - The grade will take into account the articulation between theory, algorithms, programming, simulation and creativity. All these aspects should be covered in the RL project.
  - The topics are free, it is possible to find inspiration from online resources, but the source should be cited in the introduction or sections of the project, and credits should be correctly attributed.
  - Number of pages:
    - 12 pages (excluding references) for a group of 2
    - 6 pages (excluding references) for an individual work
    - no page limit for the code
  - Deadline for handing in the project: 30.11.2025, at 23h59
  - The date for the presentation: TBD

Course details

- Main references:
  - Here is the textbook used for this lecture:
    - Sutton, R. S. and A. G. Barto, 2018, Reinforcement Learning, MIT Press
  - A pdf version of the book is available from the authors' website at:
    - http://incompleteideas.net/book/the-book-2nd.html
  - To a smaller extent, we will use:
    - Bertsekas, D., 2019, Reinforcement Learning and Optimal Control, Athena Scientific
    - Bertsekas, D., 2023, A course in Reinforcement Learning, Athena Scientific
    - Ciaburro, G, 2019, Hands-On Reinforcement Learning with R, Packt
  - Further teaching materials can be found on DS2E Seafile

Chapter 1. Introduction

Here comes a new chapter!

1. Introduction
   - 1.1 Motivation and definitions
   - 1.2 Key Elements of RL
   - 1.2 Some examples
   - 1.2.2 A repeated gift exchange game
   - 1.3 Exercises
   - 1.4 Course Outline

1. Introduction

1.1 Motivation and Key Definitions

- What is Reinforcement Learning (RL)? Why is it interesting?
- **Definitions**:

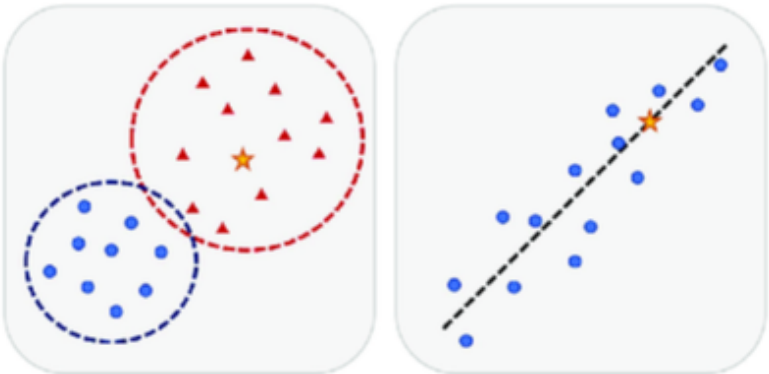
  "RL is training a computer to learn from interactions and to choose actions that are optimal for achieving a specific goal."

  "RL is a framework for learning how to interact with the environment from experience." (Brunton, 2021)

  "RL is about how an intelligent agent can learn to make sequences of decisions." (Brunskill, 2019)

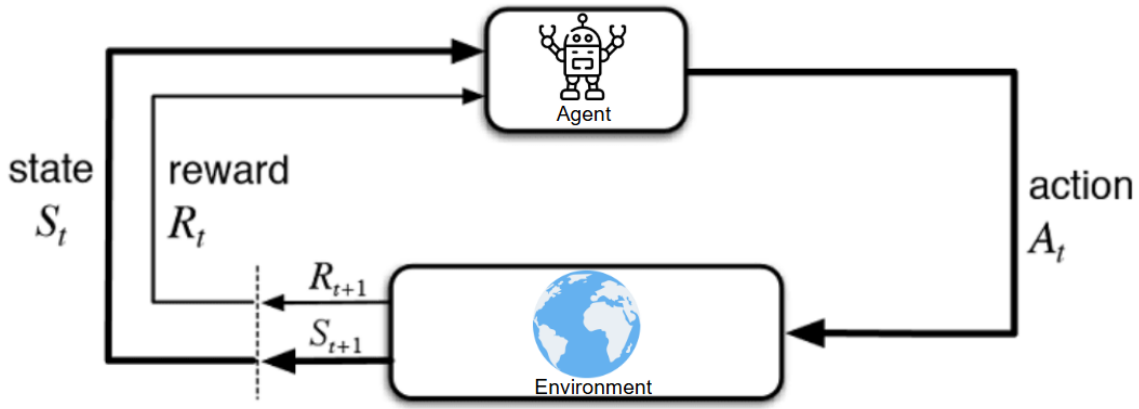
- In which respect is RL different from Supervised Learning and Unsupervised Learning?

	Supervised Learning
Data	$\{x_n, y_n\}_{n=1}^{N}$
Problem	Make forcasts (classification or prediction): If x_o is known, what is y_o ?
Learning	$y_n = f(x_n, \varepsilon_n)$ and $\widehat{y}_n = \widehat{f}(x_n), \quad \widehat{\varepsilon}_n$



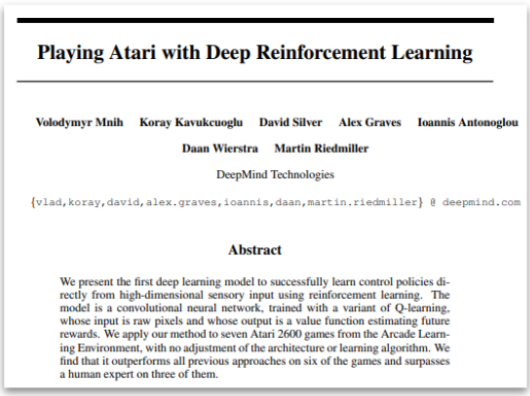
Unsupervised Learning	
Data	$\{x_n\}_{n=1}^{N}$
Problem	Looks for patterns in the data
	$\{x_n\}_{n\in S_1}, \ldots, \{x_n\}_{n\in S_G}$
Learning	Partitions the N data into G groups

	Reinforcement Learning
Data	No data required, but if available, data can be considered
Problem	Find an optimal policy which maximizes the reward.
	If the state is S_t, which action a_t shall be chosen ?
Learning	$a_t^* = f(S_t, \ldots)$ over $t = 1, ..., T$



Applications of Reinforcement Learning

- Deep Reinforcement Learning to play Atari Games in 2013
- AlphaGo Google DeepMind in 2016
- Solving a rubrik's cube with a robot hand OpenAI in 2019

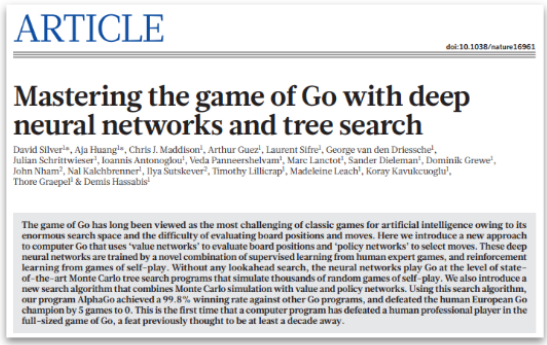


Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

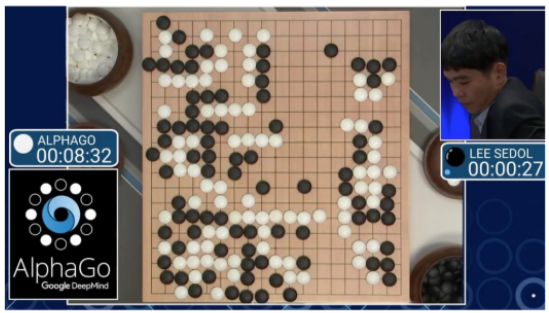


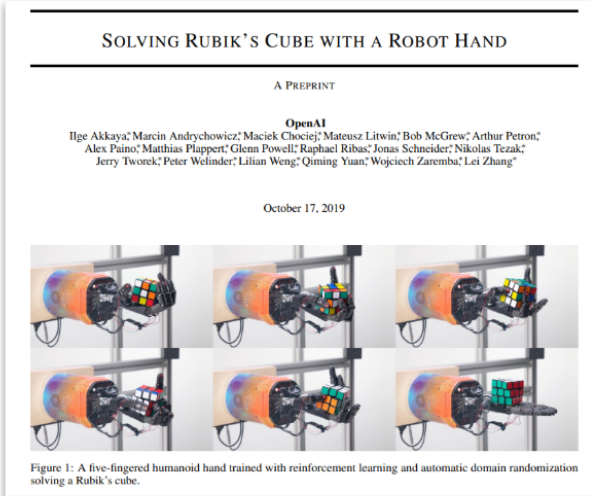
ARTICLE
doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver*, Aja Huang*, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.





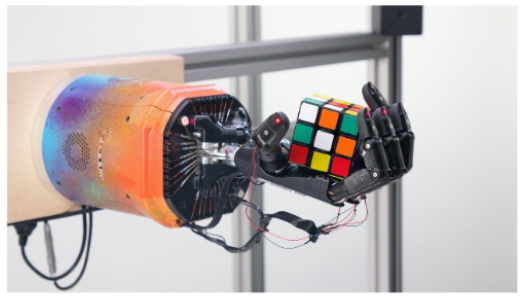
SOLVING RUBIK'S CUBE WITH A ROBOT HAND

A PREPRINT

OpenAI
Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, Lei Zhang*

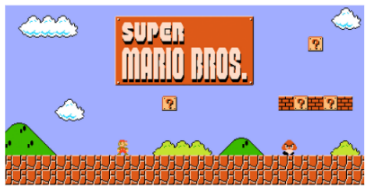
October 17, 2019

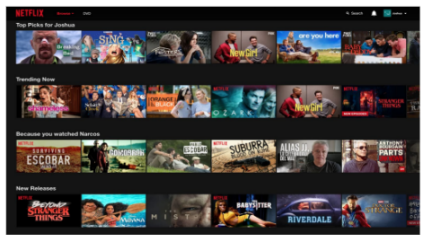
Figure 1: A five-fingered humanoid hand trained with reinforcement learning and automatic domain randomization solving a Rubik's cube.



Other Applications of Reinforcement Learning

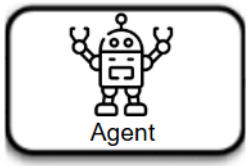
- Self-driving cars
- Cleaning robots
- Power station control
- Helicopter stunt manoeuvres
- Manage an investment portfolio
- Recommendations Systems (e.g. Netflix's)
- Assess clinical trials





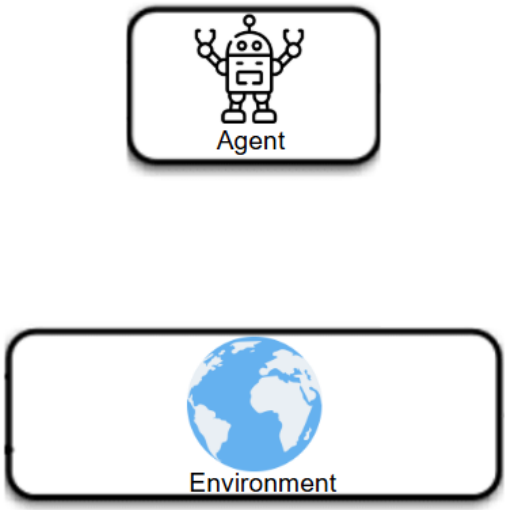
1.2 Key Elements of RL

- **Agent**: learner and decision maker

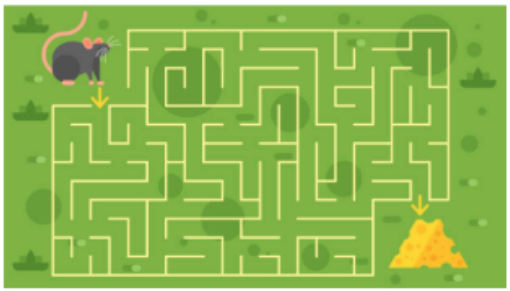


- The objective of the agent is to **maximize cumulative rewards over time**
- i.e. the agent makes sequential decisions to select actions that maximize cumulative rewards over time

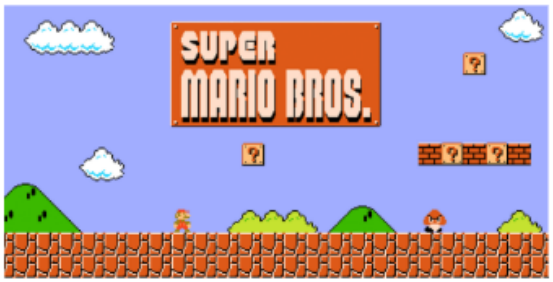
- **Environment**: the world in which the agent exists and operates



Exercise: Define the agent and environment



Agent:
Environment:

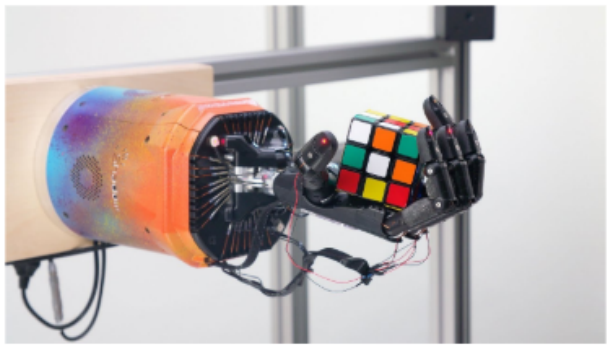


Agent:
Environment:

Agent:
Environment:

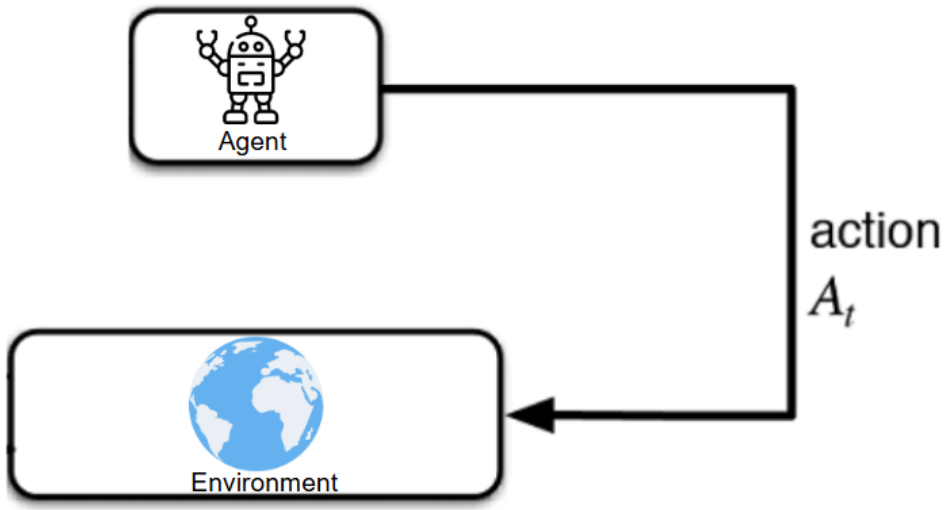
Agent:
Environment:

Agent:
Environment:

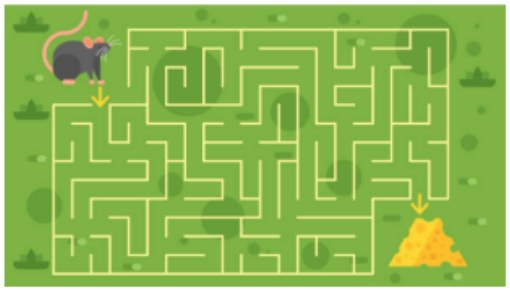


Agent:
Environment:

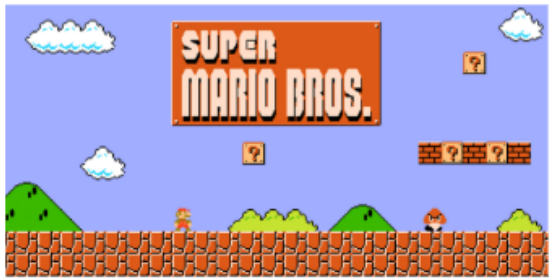
- **Action**: A decision that the agent can make in the environment



Exercise: Define actions



Actions:

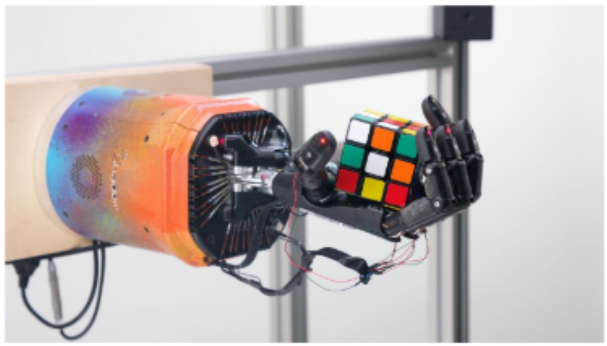


Actions:

Actions:

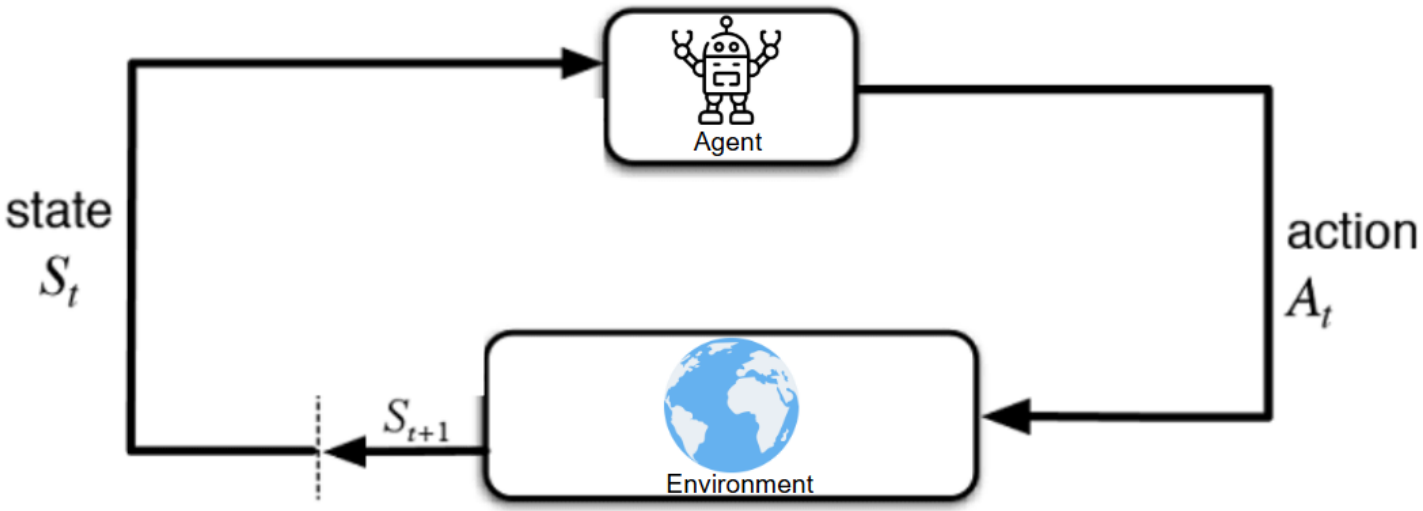
Actions:

Actions:



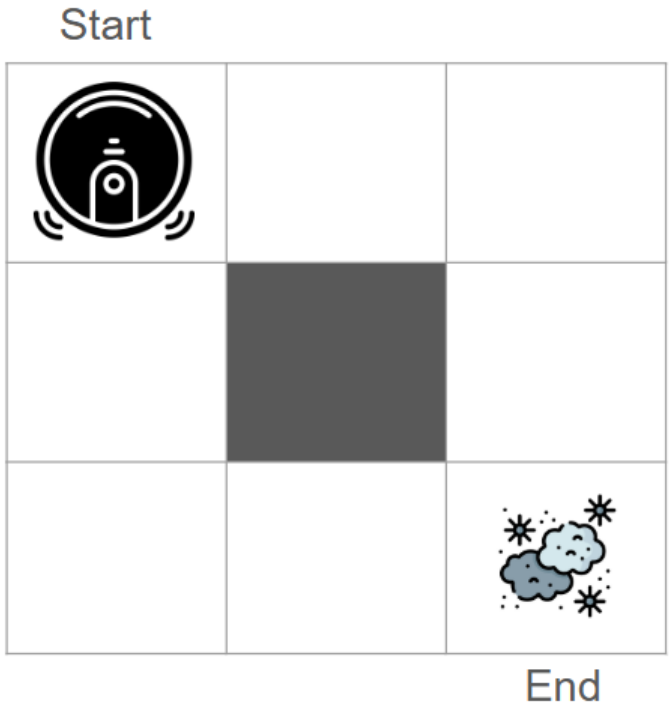
Actions:

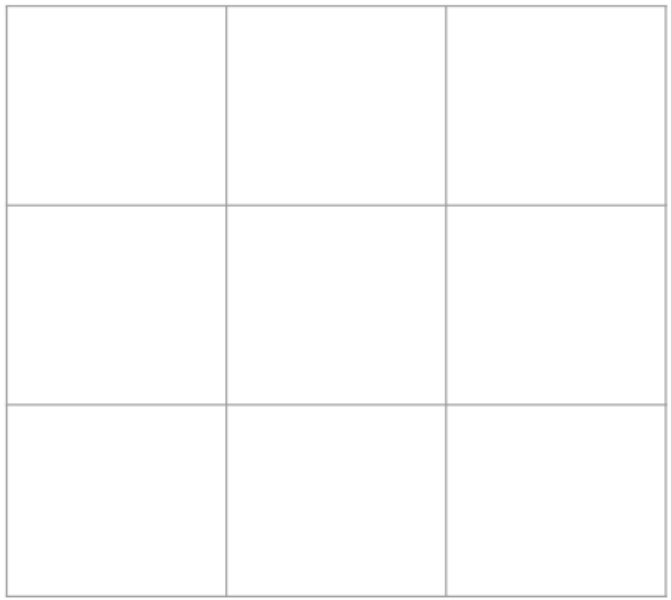
- **State**: signal given by the environment to the agent



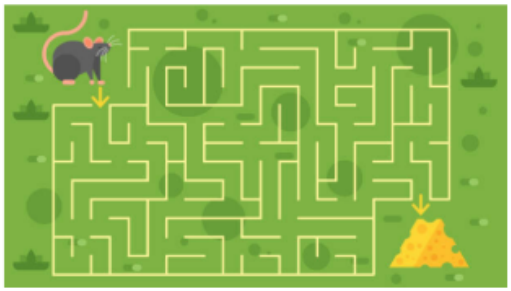
  - i.e. the information used to determine what happens next
  - i.e. the situation which the agent perceives
  - i.e. a representation of the current situation or environment that an agent uses to make decisions

Example of states in grid world

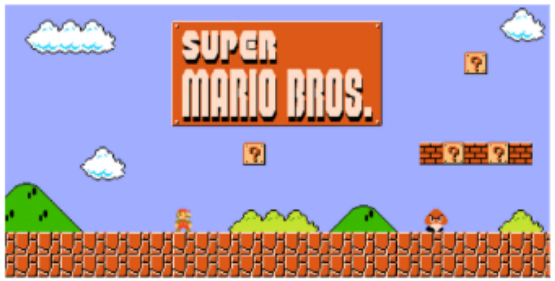




Exercise: Define states



States:

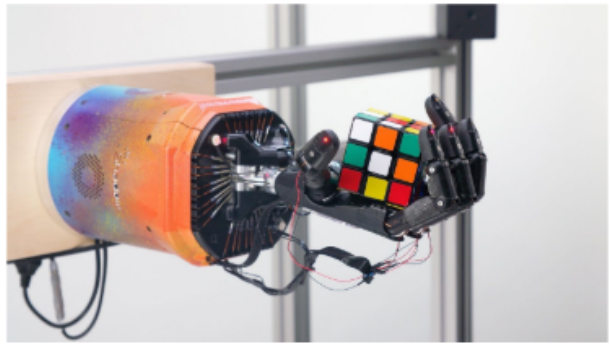


States:

States:

States:

States:

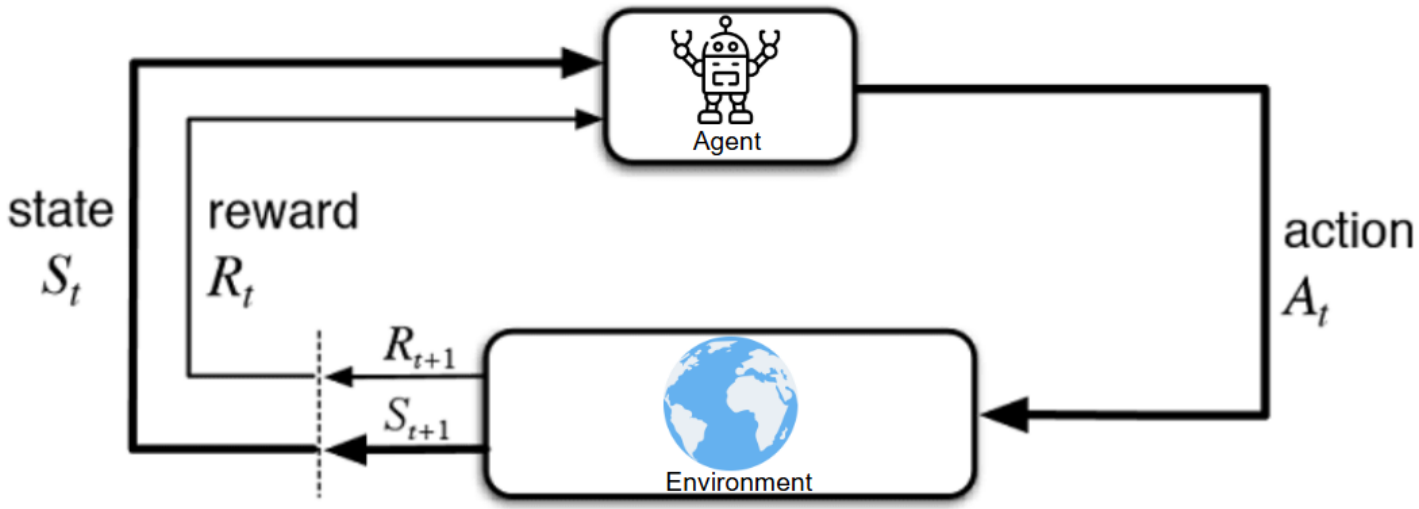


States:

Other examples of states

- Deep Reinforcement Learning to play Atari Games (2013)
  - S_t = raw pixels of the game screen (e.g., a stack of the last 4 frames) + score/lives left
- AlphaGo (Google DeepMind, 2016)
  - S_t = current board configuration (positions of black and white stones)
- Power station control
  - S_t = current energy demand, and generator status at the plant
- Helicopter stunt manoeuvres
  - S_t = helicopter's position, velocities, and wind conditions
- Manage an investment portfolio
  - S_t = current asset prices, historical returns, volatility estimates
- Assess clinical trials
  - S_t = patient's health status, treatment history, side effects, treatment progress

- **Reward**: a scalar feedback signal



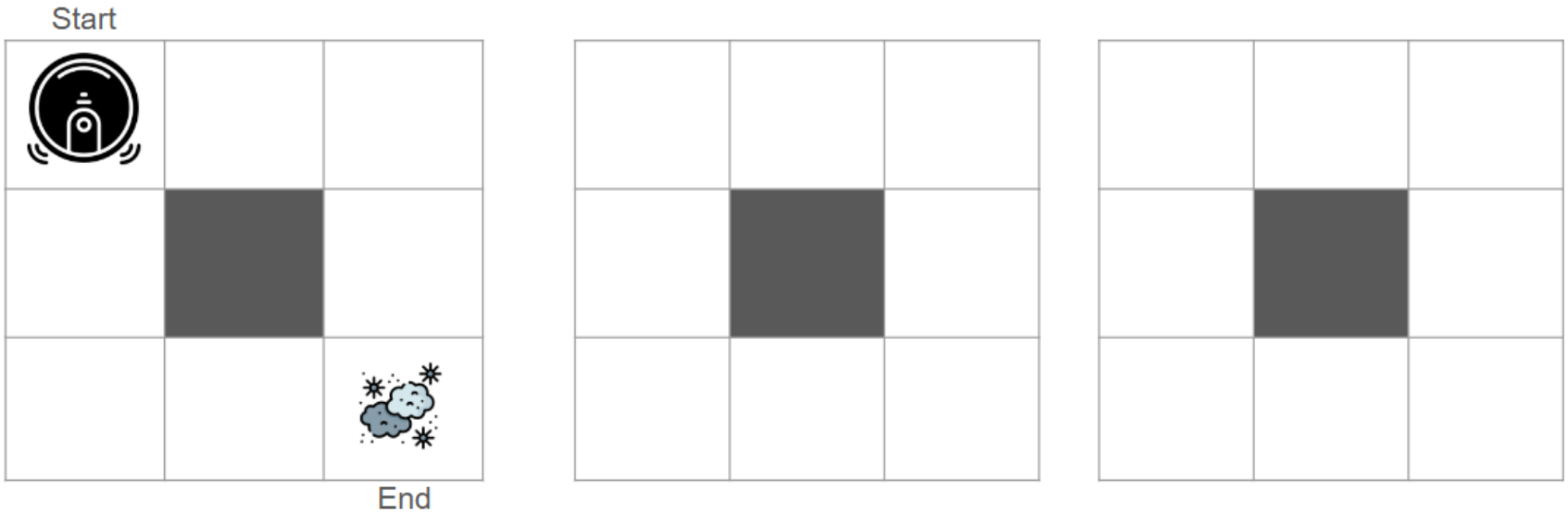
- i.e. a reward indicates how well the agent is doing at time step t
- the agent's objective is to maximize cumulative reward
- Rewards can be positive or negative

Example of Rewards

- Games (e.g. Atari, Go, Mario)
  - $+R_t$ for increasing score
  - $-R_t$ for decreasing score
- Power station control
  - $+R_t$ for producing power
  - $-R_t$ for exceeding energy demand or exceeding energy thresholds
- Helicopter stunt manoeuvres
  - $+R_t$ for following desired trajectory
  - $-R_t$ for crashing
- Manage an investment portfolio
  - $+R_t$ for each euro earning in bank account
- Assess clinical trials
  - $+R_t$ for correctly identifying effective treatments
  - $-R_t$ for suggesting ineffective treatments

- **Policy**: (denoted as π) a rule that tells the player what move to make for every state in the game
  - i.e. the decision-making engine for an agent
  - Two types of policy: deterministic and stochastic policy

Example of policies in grid world



- Deterministic policy $\pi(s) = a$
  - i.e. “In this state s, always do this action.”
- Stochastic policy $\pi(a|s) = Pr[A_t = a, S_t = s]$
  - i.e. “In this state s, choose among these actions with these probabilities.”

- **Value function** $v_\pi(s)$: the objective function to be maximized by the agent

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s \right]$$

where $0 \leq \gamma \leq 1$ is the **discount factor**.

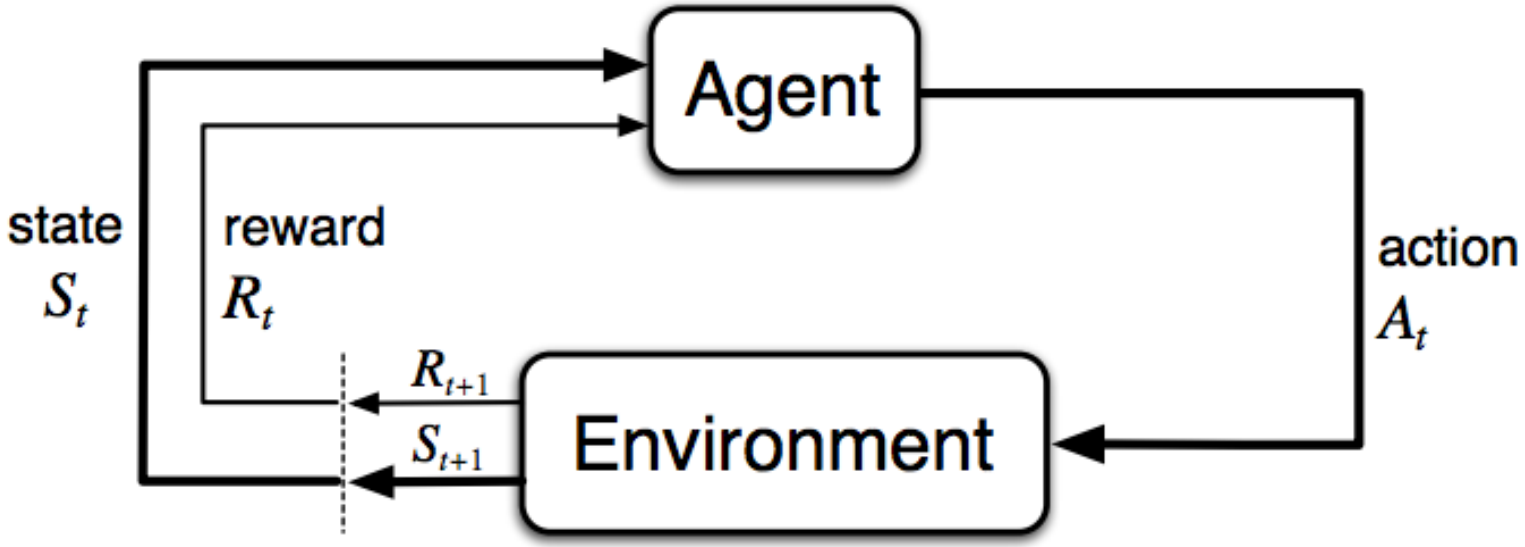
  - $\gamma = 0$: agent only cares about **immediate reward**.
  - $\gamma \to 1$: agent values **long-term rewards**.

  - i.e. $v_\pi(s)$ is used to evaluate the goodness/badness of states

Summary of RL's Key Elements:

- Agent: learner and decision maker
- Environment: what the learner interacts with (everything given to the agent and relevant for her decision)
- Action: the decision an agent makes to interact with its environment
- State: signal given by the environment to the agent about the state of the environment
- Reward: Reward's signals are given by the environment to the agent at each period t
- Policy: "a rule that tells the player what move to make for every state in the game", it generates actions
- Value function: the objective function to be maximized by the agent

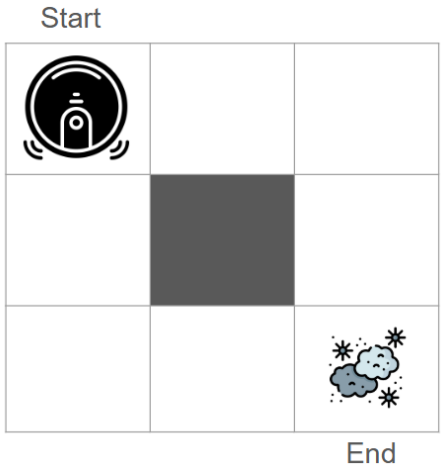
- A telling Figure



- At each loop, $R_{t+1} \leftarrow r(S_t, A_t)$ and $S_{t+1} \leftarrow s(S_t, A_t)$
- In many cases, a model of the environment is needed: describes how S changes over time as a function of (S_t, A_t)

RL General Notations

- **State:** $S_t \in \mathcal{S}$
  where $\mathcal{S}$ is the **state space**, i.e. the set of all possible states
  Example: $\mathcal{S} = \{s_0, ..., s_9\}$
- **Action:** $A_t \in \mathcal{A}(S_t)$
  where $\mathcal{A}(S_t)$ is the **action space** available in state S_t
  Example: $\mathcal{A}(s_0)$ = {up, down, left, right}
- **Reward:** $R_t \in \mathcal{R} \subseteq \mathbb{R}$
  where $\mathcal{R}$ is the set of possible rewards
  Example: $\mathcal{R} = \{+1, 0, -1\}$



Key Ideas of RL: Exploration and Exploitation

Silver (2015)

- RL is like trial-and-error learning
- The agent should discover a good policy from its experiences of the environment
- Without losing too much reward along the way

Silver (2015)

- **Exploration** = finds more information about the environment
- **Exploitation** = exploits known information to maximize reward
- It is usually important to explore as well as exploit

Example of Exploration and Exploitation

- Case study: Recommendation Systems
- Action (i.e. suggesting a movie) $A_t \in \{a_1, a_2, a_3, a_4\}$
- Reward $R_t = +1$ if the user watch it. $R_t = -1$ otherwise

a_1

a_2

a_3

a_4

- **Exploitation**: Keep recommending a_1 because it has given the highest click/watch probability so far
- **Exploration**: Recommend a_2 or a_3

The goal is to strike a balance between exploration and exploitation because

- Too much exploitation: user sees a_1 recommended every day (boring, stuck in a loop)
- Too much exploration: user gets random suggestions (romantic comedies, documentaries), many of which they dislike

In which respect is RL different from econometrics?

- We can easily compare both topics if we reparameterize the model as follow $a \to x$, $S \to y$ and introduce a vector of parameters β.
- The objective in econometrics is parameter estimation. Given some assumptions on the random variables Y, and the DGP generating the observations y_n, x_n, to choose β such that:

$$\max_{\beta} Q\left(\left\{V\left(\beta; x_n, y_n\right)\right\}_{n=1}^{N}\right),$$

where Q correspond to a log-likelihood function or a (minus) sum a squared function (or any alternative loss function).

- The objective in RL, is to determine the actions y in a environment described by x, β such that:

$$\max_{y} V(\beta; x_n, y_n)$$

- Just as in econometric, given x_n, the value of y_n is often influenced by a random term.
- In this lecture, the notations are not the same as in econometrics: actions are denoted a, and the environment by S and so

$$\max_{a} V(S; a)$$

- This discussion illustrates that RL is closer (in spirit) to microeconomics or decision theory than to econometrics, but the tools used for optimization are quite the same.

- The context of RL is often defined by repeated interactions between agent and the environment
- At each time or step t, the agent observes the environment state $S_t \in \mathcal{S}$, and the agent then selects an action $A_t \in \mathcal{A}(S_t)$
  - $\mathcal{S}$ is the set of possible states
  - $\mathcal{A}(S_t)$ is set of actions available in state S_t
- The agent receives a reward, $R_t \in \mathcal{R} \subset \mathbb{R}$, such that

$$R_t = r(A_t, S_t, ...)$$

- One time or step later, $t \leftarrow t + 1$, the new state is $S_{t+1} = s(A_t, S_t, ...)$ and the loop continues as in step t.
- The reward and the evolution of the environment may include some randomness
- Values are to be estimated and re-estimated

- At step t,
  - The agent:
    - Receives state S_t
    - Executes action A_t
    - Receives a reward R_t for the period t
  - The environment:
    - Emits state $S_t = s(S_{t-1}, A_{t-1}, ...)$
    - Receives action A_t
    - Emits a reward $R_t = r(A_t, S_t, ...)$
    - The next step $t \leftarrow t + 1$
    - Emits state $S_{t+1} = s(S_t, A_t, ...)$
- This RL is flexible enough to allow
  - the agent to be a computer, a human or an animal
  - the environment to be a computer, a human or an animal
  - inclusion of several agents interacting with each other

1.2 Some examples

1.2.1 A repeated investment problem

- From microeconomics... Each period, an investor receives a reward (monetary payoff) from investing a:

$$r(a, S) = S\sqrt{a} - a \tag{1}$$

  and the state evolves according to $S_{t+1} = S_t + a_t$. Usually there is some randomness to be included in investment decisions: $S_{t+1} = S_t + a_t + \varepsilon_t$.
- Let us think about the issues and difficulties arising when we want to solve such a problem.

1.2 Some examples

1.2.2 A repeated gift exchange game

- A game with strategic interaction, adapted from Charness et al. (2012)
- There are actually two agents interacting. Without loss of generality, Agent 2 is included in the ”environment of Agent 1”, so that the LR-setup is satisfied
- Each period, the employer receives a reward (monetary payoff) which depends upon her action a:

$$r_1(a, S) = (240 - a)S \tag{2}$$

- The employee (environment) also receives a reward and chooses S:

$$r_2(a, S) = a - c(S) - 20, \tag{3}$$

- Draw the ”constant-reward curves”, the extensive form of the game, and discuss about the Nash equilibrium and the social optimum.

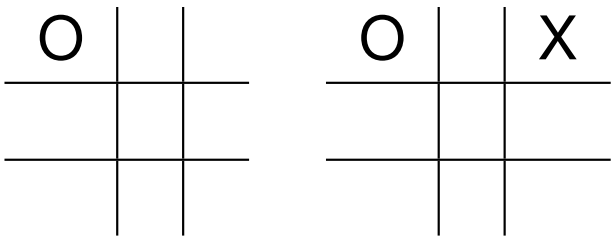
- Can the problem be restated in the RL language?
  - $\mathcal{A} = \{a : 20 \leq a \leq 120\}$.
  - The environment: $\mathcal{S} = \{s : 0 \leq s \leq 1\}$. At $t + 1 \geq 1$ there may be a probability distribution depending upon past actions: $Pr[S = s|a_t, S_t]$
  - Agent 1's environment includes in this example another agent which determines how the state changes over time.

- If the employer maximizes her profit for given effort level, she would set $a = 20$ the minimum wage.
- If the worker (in the environment of Agent 1) is maximizing r_2 for a given level of a would choose $s(a) = 0.1$.
- This strategy yields $r_1(20, 0.1) = 22$ and $r_2(20, 0.1) = 0$ and is clearly not Pareto-optimal since (for instance) $r_1(60, 0.5) = 90$ and $r_2(60, 0.5) = 24$.
- Draw the extensive form of the game.
- What would happen if the agent is not only sensitive to the monetary payoff ? The agent may be altruistic, fair, and be sensitive to equity...
- https://en.wikipedia.org/wiki/Prisoner's_dilemma

1.2 Some examples

1.2.3 Tic-Tac-Toe

- Tic-Tac-Toe is a quite popular game:



- The RL view of this game...
  - the agent is the computer (plays with X)
  - $\mathcal{A}_t = \{$the coordinates i, j of $X[i, j] : i, j$ are admissible$\}$.
  - The set of states: $\mathcal{S} = \{3^9$ different boards$\}$.
  - At t the agent has to choose $A \in \mathcal{A}_t$ in order to try to win the game.

- ...The RL view of this game
  - The agent's environment includes in this example another player who contributes to change the state $S_{t+1} = s(S_t, B_t, a)$ by her choice of B_t.
  - The reward could be +1 for winning the game, 0 for a tie, and -1 for loosing the game
  - Discussion: how to attribute values to different states?

$t = 2$

O		X
	O	

$t = 3$

O		X
	O	
O		X

$t' = 3$

O	X	X
	O	
O		X

$t'' = 3$

O		X
	O	X
O		X

- Starting from S_t, RL assigns to each of the possible 3^{9-2t} future states, the probability to win from this state.
- How to determine these values (these probabilities)? One main issue in RL is the evaluation and the updating of these values.
  - The computer could be used to play the game (against itself) and learn from playing about these values (the winning probabilities)
  - While playing, new information on the values of the different states is revealed, and it is important (in order to maximize the value) to update the older state's value $V(S_t)$ estimate, using the information provided by the new estimate of this value.
  - $V(S_t) \leftarrow V(S_t) + \alpha(V(S_{t+1}) - V(S_t))$, where $\alpha > 0$ but small is the **step-size parameter**.
  - This updating rule, based on the difference $V(S_{t+1}) - V(S_t)$ is known as **temporal difference**.

- Exercise: compare this updating formula with one used by algorithms solving nonlinear models
- In order to have a value for many states S_t, it is necessary not always choosing the action with highest value. In RL, actions have two (conflicting) aims: to maximize V_t and to obtain information about the different V_s.
- **Greedy** action: the action which leads to the state with highest value.
- **Exploratory** action: defined in opposition to greedy action. It allows to evaluate or reevaluate the value over states that are not well known.
- How to characterize the optimal trade-off between exploitation and exploration?

- Several python or R codes are available on the web:
  https://sunjackson.github.io/2017/03/07/9fc8a904da793269aaa4c4c8a6392034/
  https://medium.com/byte-tales/the-classic-tic-tac-toe-game-in-python-3-1427c68b8874
- Applications to game theory ca be found there:
  https://axelrod.readthedocs.io/en/stable/index.html

1.3 Exercises

- Exercise: Formalize the road closure problem using the language of RL.
  - Define the states, actions, rewards, and environment
  - Propose an algorithm that would allow an agent to avoid closed roads

- Exercise: Read and explain the code available at https://sunjackson.github.io/2017/03/07/9fc8a904da793269aaa4c4c8a6392034/ Do you agree with K. Jacobs writing on his blog that ”Forward sampling ... does not require any learning”?
- Exercise: Solve the exercises of Sutton and Barto (2018, p.12-13).

Course Outline

This course follows the structure of Sutton, R. S. and A. G. Barto (2018)

- Chapter 1: Introduction
- Chapter 2: Multi-Arm Bandit
  - (e.g. slot machines, Netflix's recommendation systems, applications in economics)
- Chapter 3: Markov Decision Processes
  - (e.g. Recycling robots)
- Chapter 4: Dynamic Programming
  - (e.g. Grid World / Maze Solving)
- Chapter 5: Monte Carlo Methods
- Chapter 6: Temporal-Difference Learning
  - SARSA and Q-Learning, and Deep Q-Learning
  - (e.g. Atari games, self-driving cars, early versions of AlphaGo, solving Chess)
- Recap and exam preparation