REINFORCEMENT LEARNING

Seminar @ UPC TelecomBCN Barcelona (2nd edition). Spring 2020.

Instructors

Josep Vidal

Margarita Cabrera

Xavier Giró-i-Nieto

Organizers

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

telecom BCN

Supporters

Google Cloud

GitHub Education

+ info: http://bit.ly/upcrl-2020

telecos BCN

UPC

Plan of the seminar

Schedule: Fridays from 11:00 to 13:00, from 21st Feb to 28th Apr.

Oral presentation of works on 8th May.

Lecturers: Marga Cabrera, Xavier Giró, Josep Vidal

Class notes: Get them in Atenea

Lab: Bring your own laptop,

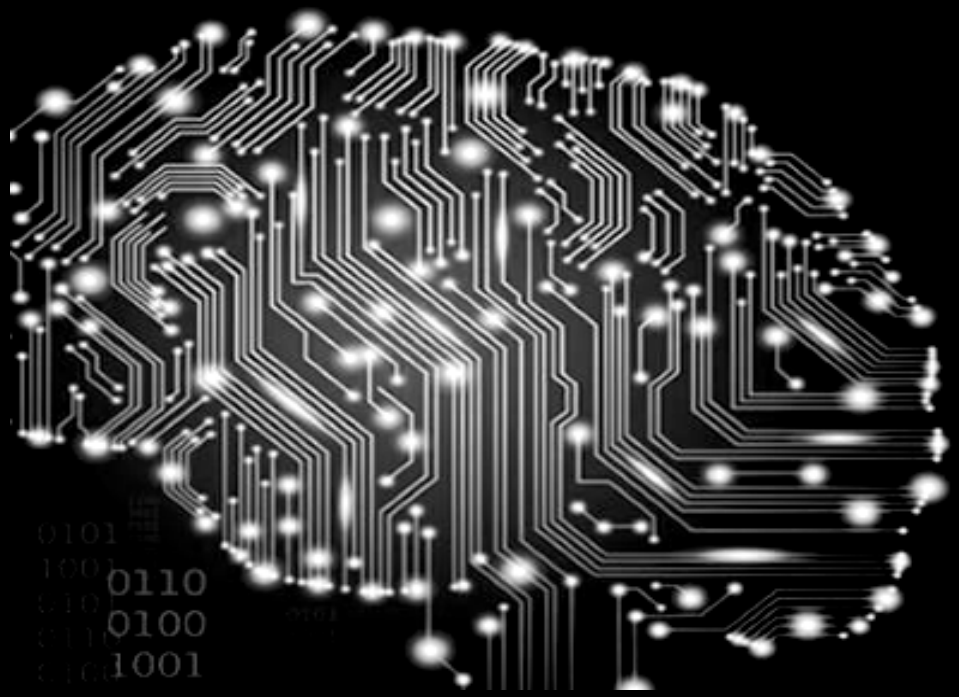
- Chapters 2-5: use Matlab or Python
- Chapters 7-8: Python on Google's Colab environment

Daily deliverable: Quiz and practice assignments

Evaluation: Deliverables and two mini-projects

Textbook: Sutton & Barto

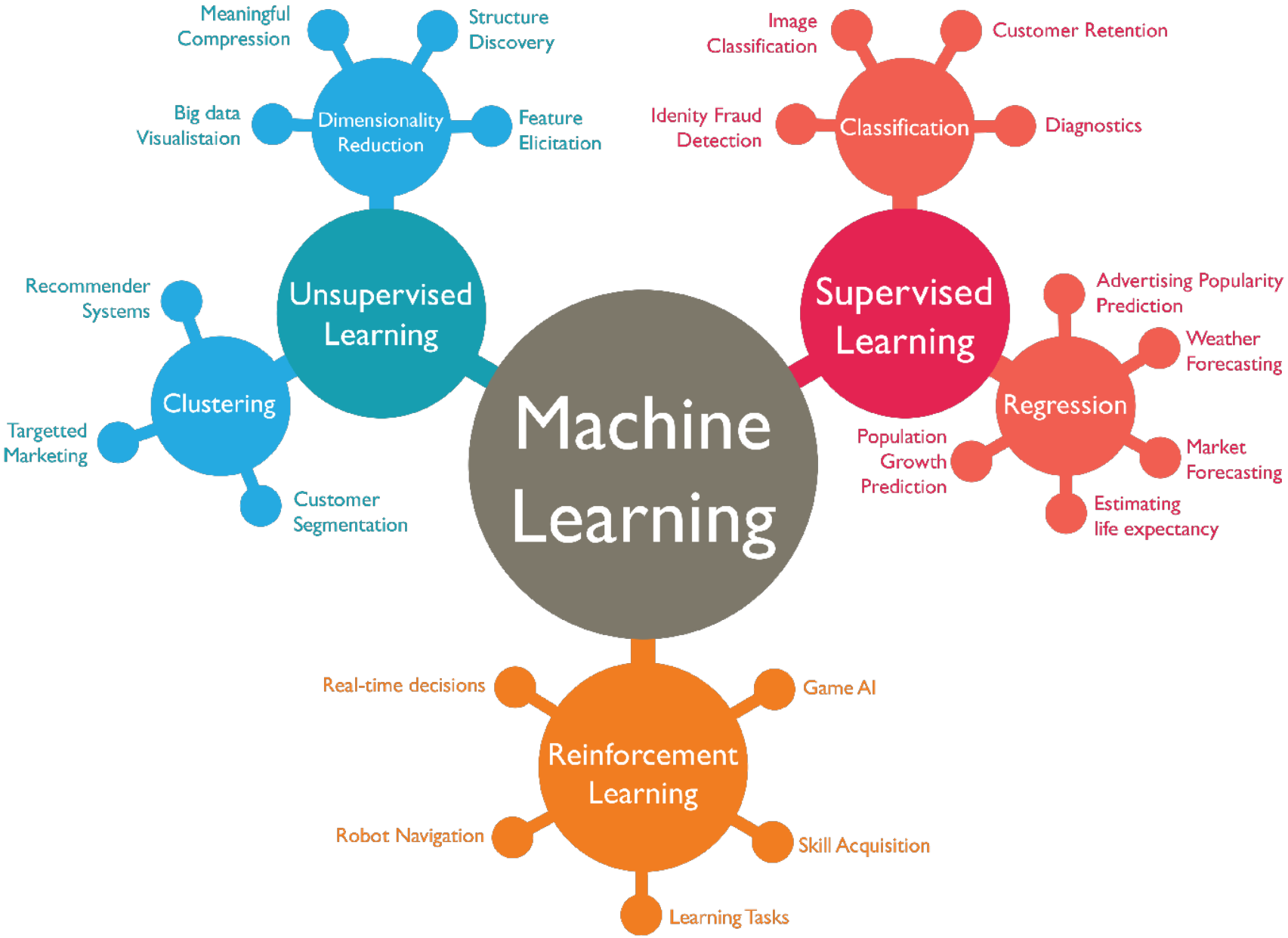
1. Introduction to Reinforcement Learning



Why do we need machines taking decisions?

Fundamental causes of human irrational behavior

- Evolutionary causes: fear, shyness, conformism, shame,…
- Physiological causes: when we learn, some neurons connect but some other existing connections weaken
- Mental sloth
- Unability to manage many variables simultaneously
- Unability to handle basic concepts of probability calculus
- Selfish bias for fear of losing self-steem


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

A taxonomy of machine learning…

We want to build a machine that learns how to take decisions out of sensorial data (feature vectors) $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, …

There are three main approaches:

1. **Supervised learning**: the learning rule is provided with the desired outputs y_1, y_2, y_3, y_4 , … for each given input. The objective is generating the right output when fed with a new input.

2. **Unsupervised learning**: a model is built for the values of $\mathbf{x}$ that can be used to explain some phenomenon, to understand the structure of data, to predict, to communicate, etc.

3. **Reinforcement learning**: we do not have reference answers. The machine takes some actions a_1, a_2, … affecting the environment, and gets some reward (or punishment) r_1, r_2, … The objective is learning to act in a way that long term rewards are maximized.

Supervised learning

Data base

- Feature vectors (observations): $\mathbf{x}_i \quad i = 1, \ldots, N$
- Each belonging to one of the c classes (nature states): y_i

Objective

Design a procedure that learns to associate $\mathbf{x}_i$ and y_i through the function $h_{\boldsymbol{\theta}}$

$$\hat{y}_j = h_{\boldsymbol{\theta}}(\mathbf{x}_j)$$

that depends on some parameters $\boldsymbol{\theta}$, using the data base and optimising some loss function

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L\left(y, h_{\boldsymbol{\theta}}(\mathbf{x})\right)$$

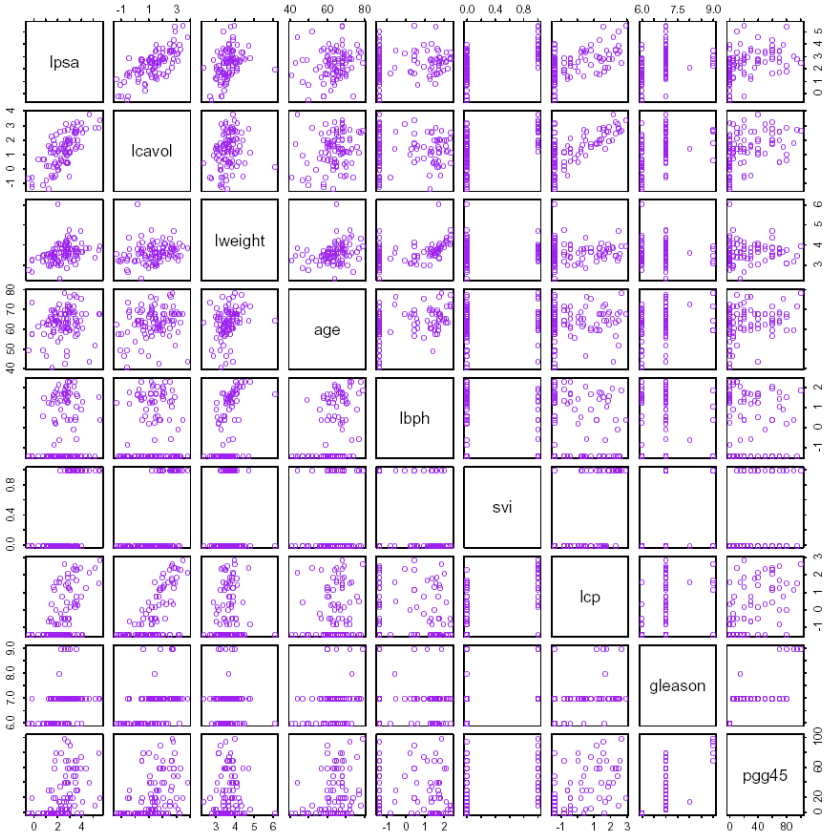
Example 1.1. Diagnose of prostate cancer

Build a data base of clinical histories $\mathbf{x}_i$ containing relevant information

- lpsa
- log cancer volume (lcavol)
- log prostate weight (lweight)
- age
- log benign prostatic hyperplasia (lbph)
- seminal vesicle invasion (svi)
- Gleason score (gleason)
- percent of Gleason scores 4 or 5 (pgg45)

of healthy and sick patients (y_i), and build a machine able to predict prostate cancer condition

$$\hat{y}_j = h_{\theta}(\mathbf{x}_j)$$



A taxonomy of machine learning…

We want to build a machine that learns how to take decisions out of sensorial data (feature vectors) $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, …

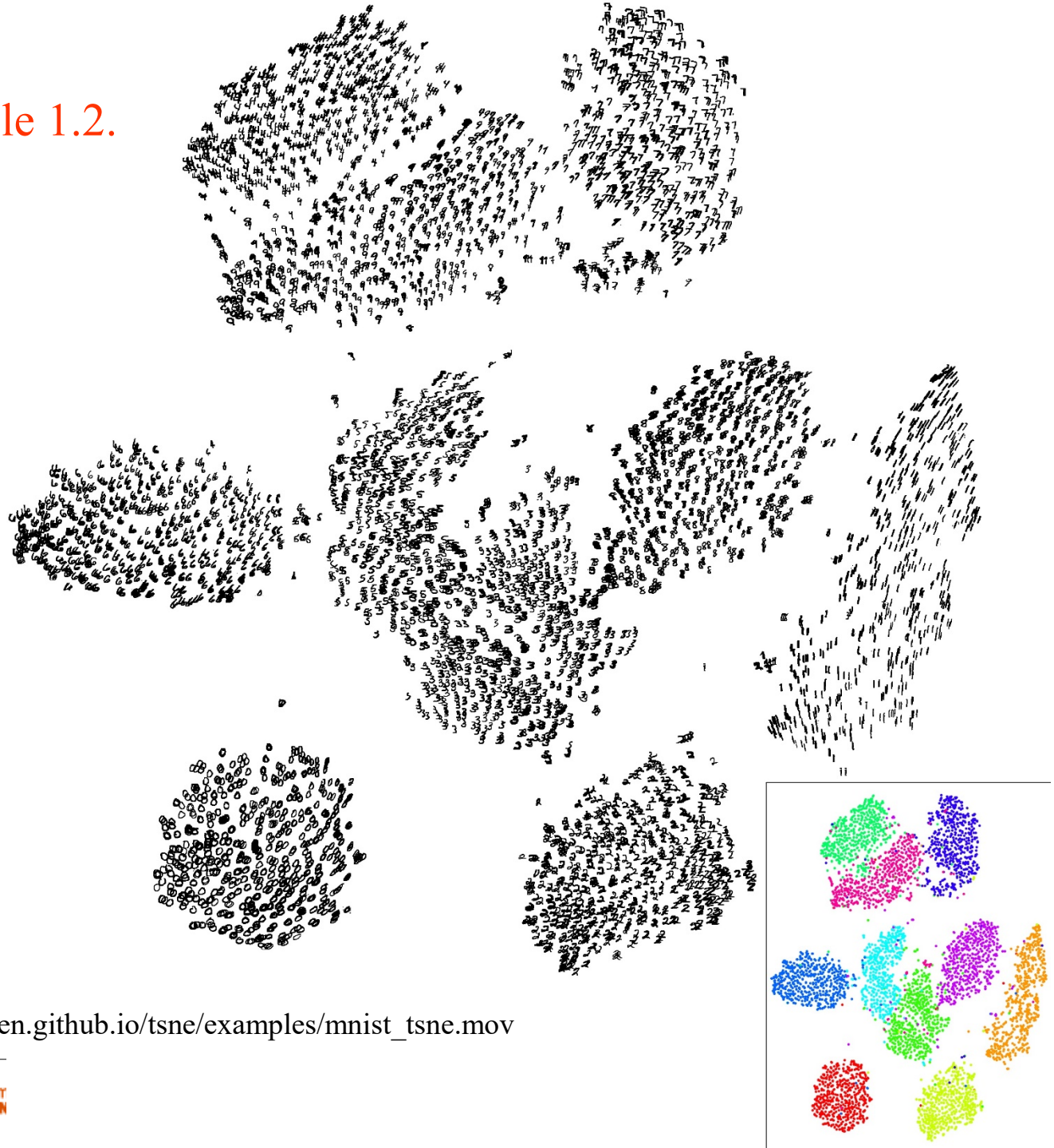
There are three main approaches:

1. **Supervised learning**: the learning rule is provided with the desired outputs y_1, y_2, y_3, y_4 , … for each given input. The objective is generating the right output when fed with a new input.

2. **Unsupervised learning**: a model is built for the values of **x** that can be used to explain some phenomenon, to understand the structure of data, to predict, to communicate, etc.

3. **Reinforcement learning**: we do not have reference answers. The machine takes some actions a_1, a_2, … affecting the environment, and gets some reward (or punishment) r_1, r_2, … The objective is learning to act in a way that long term rewards are maximized.

Example 1.2.



https://lvdmaaten.github.io/tsne/examples/mnist_tsne.mov

A taxonomy of machine learning…

We want to build a machine that learns how to take decisions out of sensorial data (feature vectors) $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, …

There are three main approaches:

1. **Supervised learning**: the learning rule is provided with the desired outputs y_1, y_2, y_3, y_4 , … for each given input. The objective is generating the right output when fed with a new input.

2. **Unsupervised learning**: a model is built for the values of $\mathbf{x}$ that can be used to explain some phenomenon, to understand the structure of data, to predict, to communicate, etc.

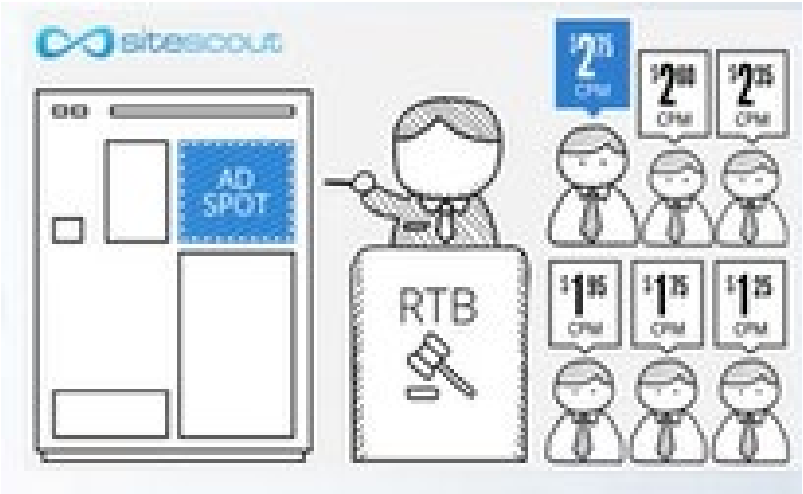
3. **Reinforcement learning**: we do not have reference answers. The machine takes some actions a_1, a_2, … affecting the environment, and gets some reward (or punishment) r_1, r_2, … The objective is learning to act in a way that long term rewards are maximized.

Example 1.3. Manage advertisements in a website (YouTube, Facebook,…)

Each user visiting the page is shown a banner that he/she may either click or not.

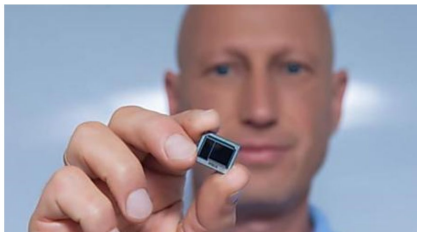
On the other side we have banner ad providers: companies that will pay us if users click on their banners.

Pick the banner we want to show,… but we cannot use supervised learning because we have no data set nor previous experience.



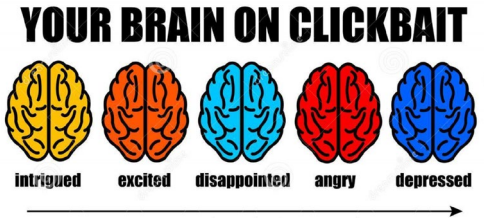
How to solve it?

A general idea: initialize with a naive solution and try to learn from trial and error.



If your computer has an Internet connection, this is something you should to today

Issue 1. Introduce optimality: maximize benefit right now vs. make users happy so that they return.



Issue 2. Follow the current strategy or discover better ones?

Comparison…

Supervised learning	Reinforcement learning
Learning to approximate reference answers	Learning optimal strategy by trial and error
Needs correct answers	Needs feedback on agent's own actions
Model does not affect the input data	Agent can affect its own observations
	Training can be very costly, unless we have computer models we can rely on

Wayve, "The first example of deep reinforcement learning on-board an autonomous car" (2018)

Some examples of RL applications

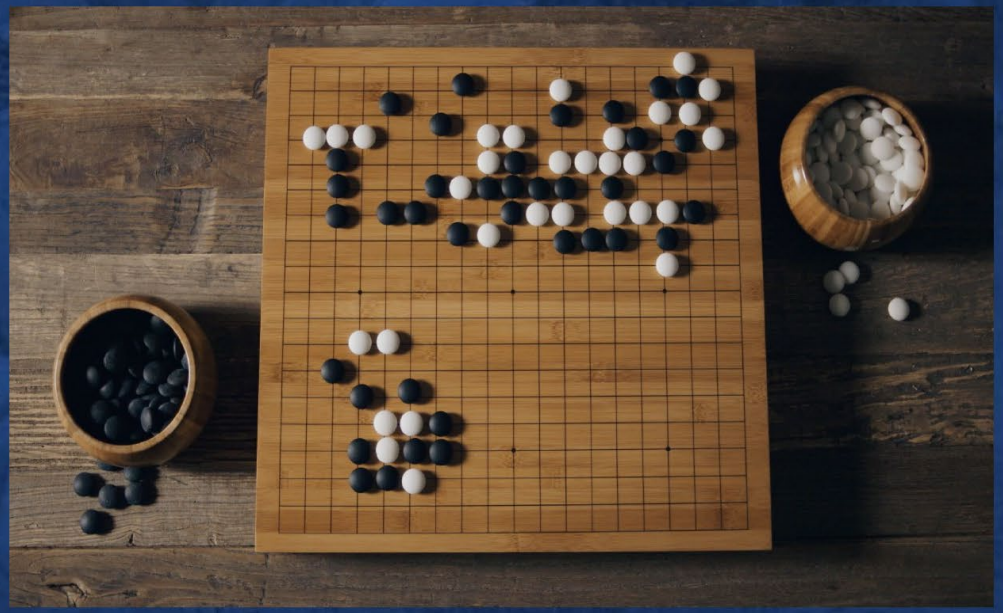
- Self-driving vehicles (cars, helicopters)
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Recommendation system for online store
- Find pages relevant to queries
- Defeat the world champion at Backgammon, Go, Chess

Why is Go hard for computers to play?

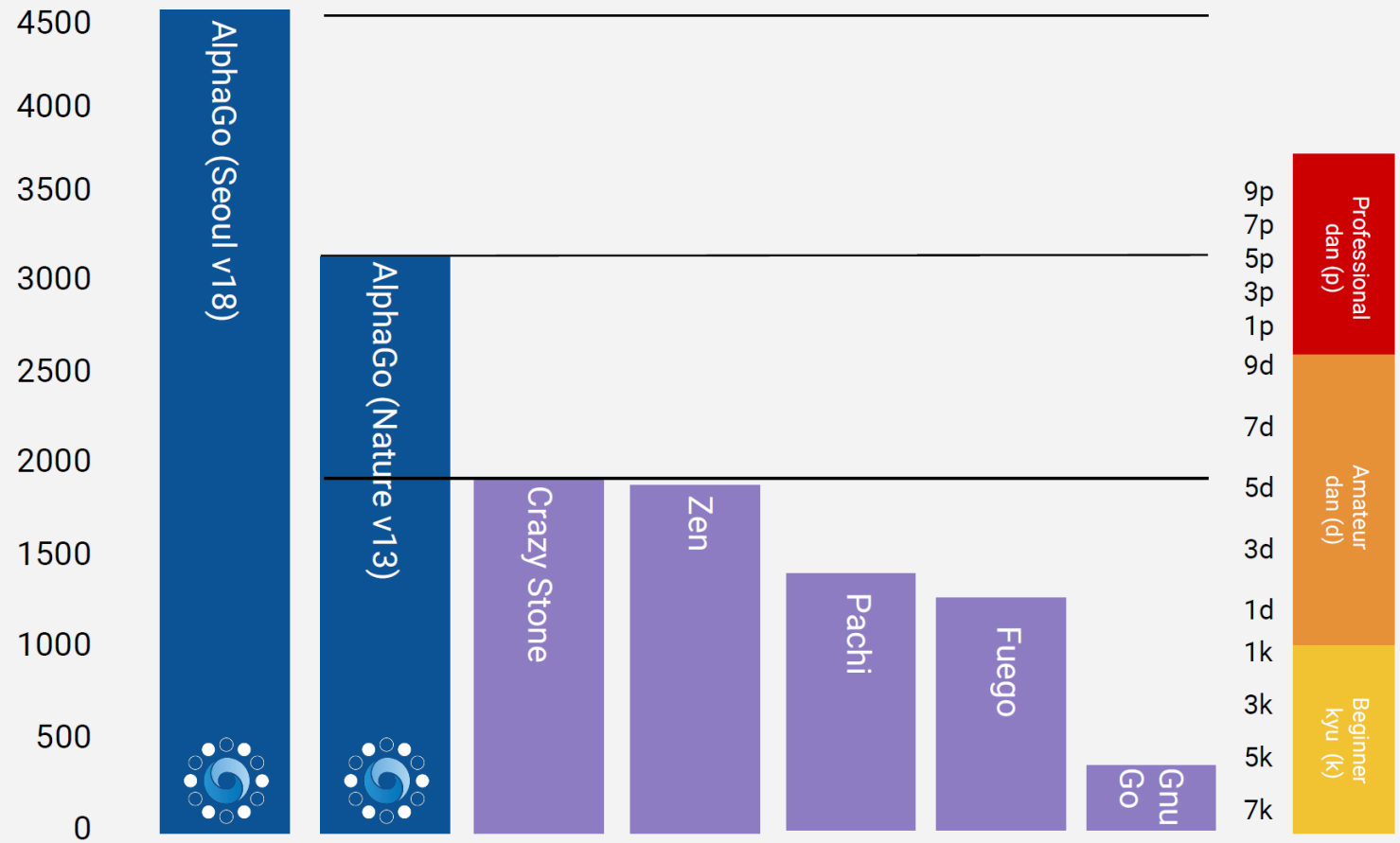
Game tree complexity = b^d

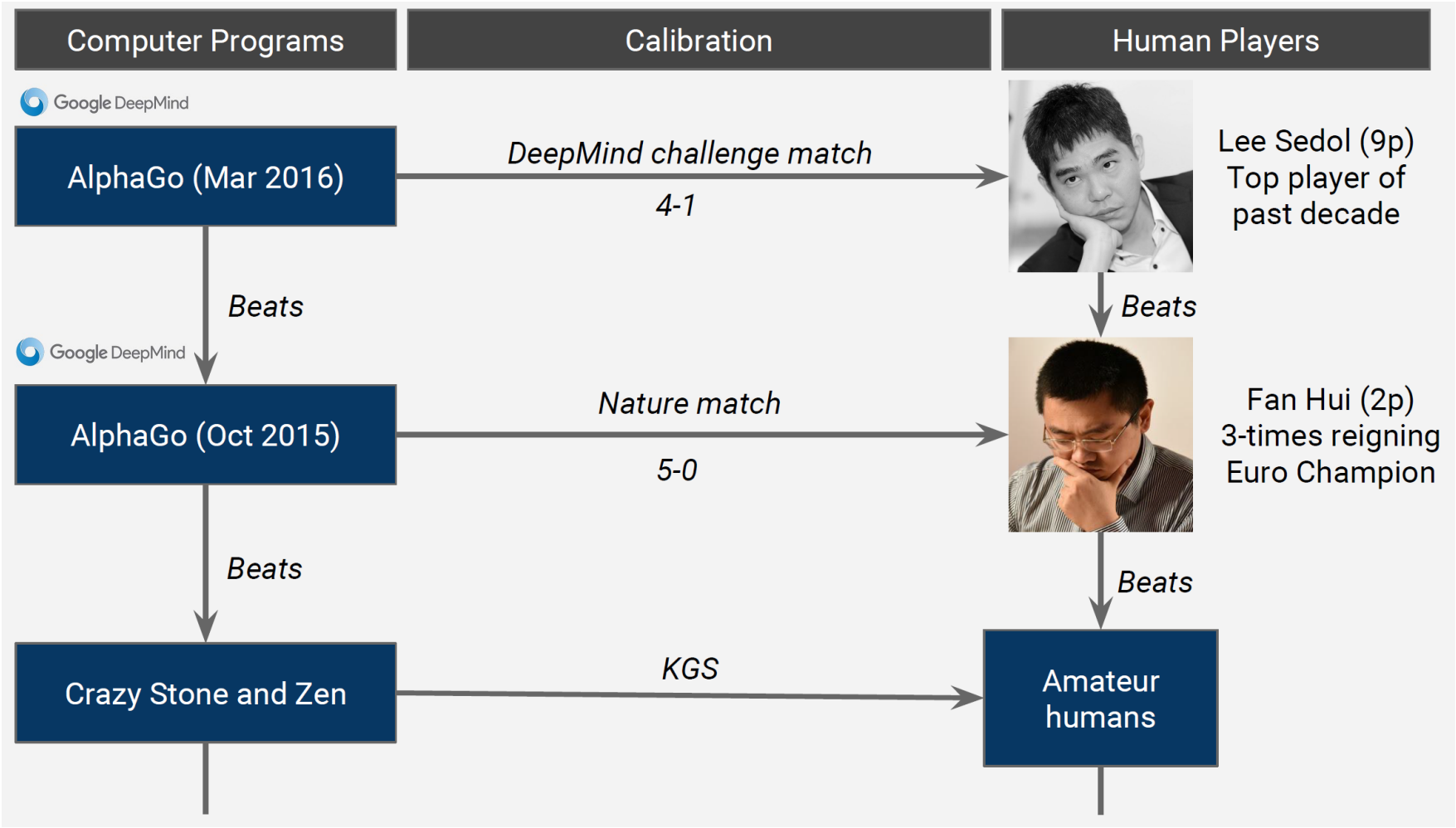
Brute force search intractable:

1. Search space is huge
2. “Impossible” for computers to evaluate who is winning



Evaluating AlphaGo against computers



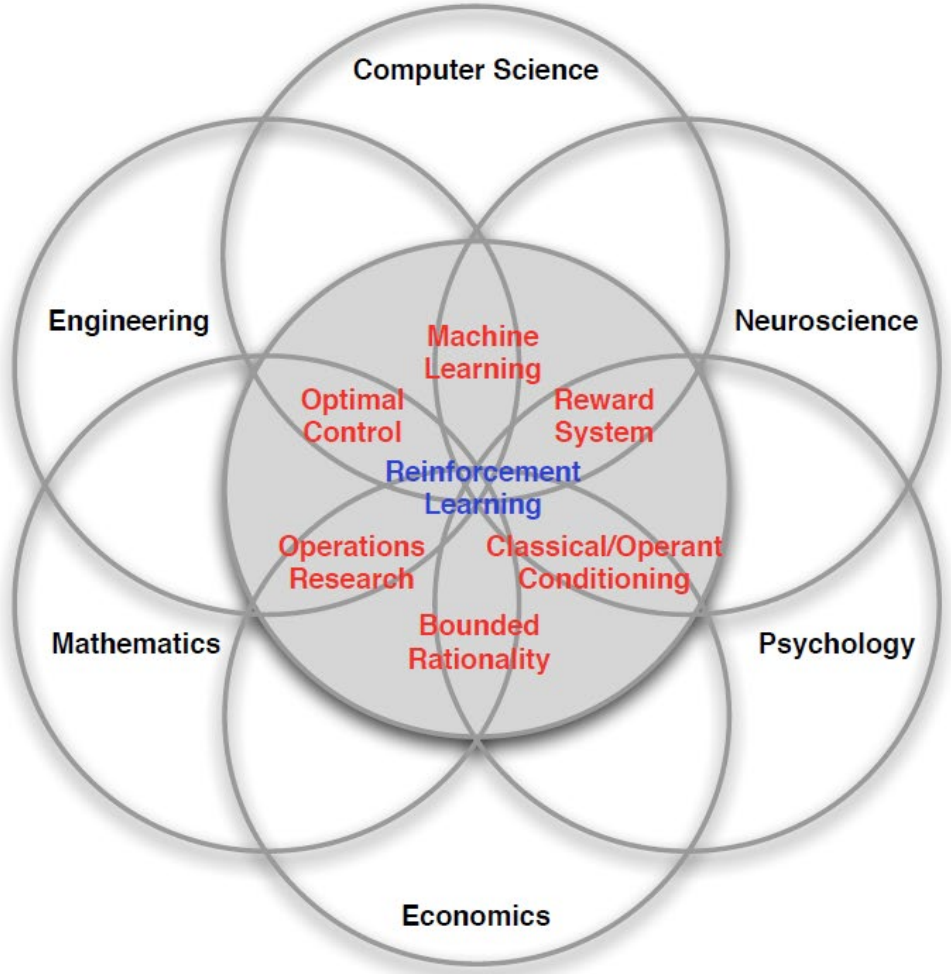


DeepMind was bought by Google for >$500 millions in 2014.
The story behind it https://www.youtube.com/watch?v=WXuK6gekU1Y

Some examples of RL applications

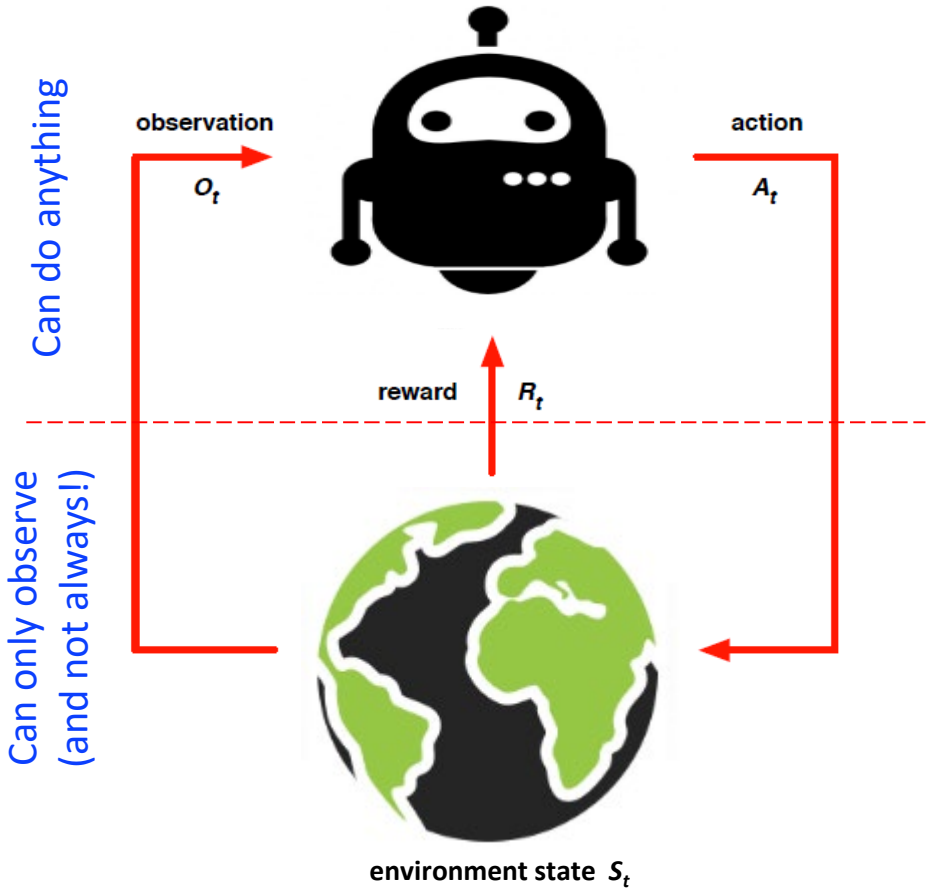
- Conversation systems: learn to make user happy
- Quantitative finance, portfolio management
- Deep learning: optimizing non-differentiable loss, finding optimal architecture
- Play Atari games better than humans

Many facets of reinforcement learning



Problem statement

Goal-directed agent interacting with an uncertain environment



At each step t, the agent:

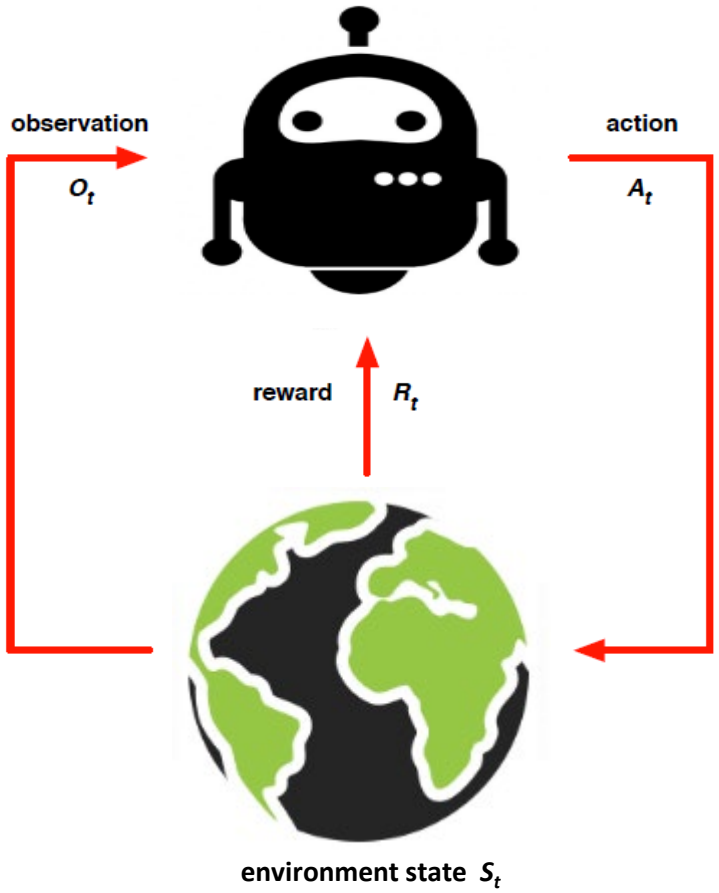
- Executes action A_t
- Receives observation O_t
- Receives scalar reward R_t

The environment:

- Receives action A_t
- Emits observation O_{t+1}
- Emits scalar reward R_{t+1}

t increases at each environment step

Back to example 1.3. Manage advertisements in a website (YouTube, Facebook,…)



Action A_t : show a certain banner

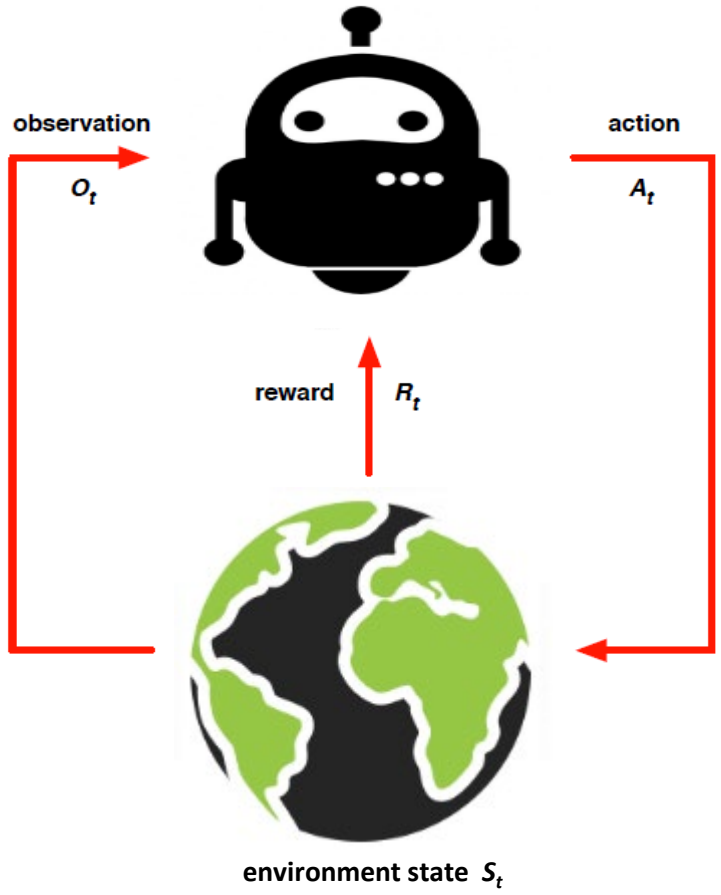
Observation O_t : number of clicks by users

State S_t : equal to O_t in a fully observable environment. Maybe discrete or continous.

Reward R_t : reward from advertising companies (maybe a delayed signal)

Can actions modify the environment? For example, showing "clickbait" banners will increase click rates, but after a while no one will trust you!

Markov Decision Process (MDP)



Formally defined as:

State S_t : takes values $s \in \mathcal{S}$

Action A_t : takes values $a \in \mathcal{A}$

Reward R_t : takes values $r \in \mathcal{R}$

History: the sequence of S, R, A

$$H_t = S_1 R_1 A_1, \ldots, A_{t-1} S_t R_t$$

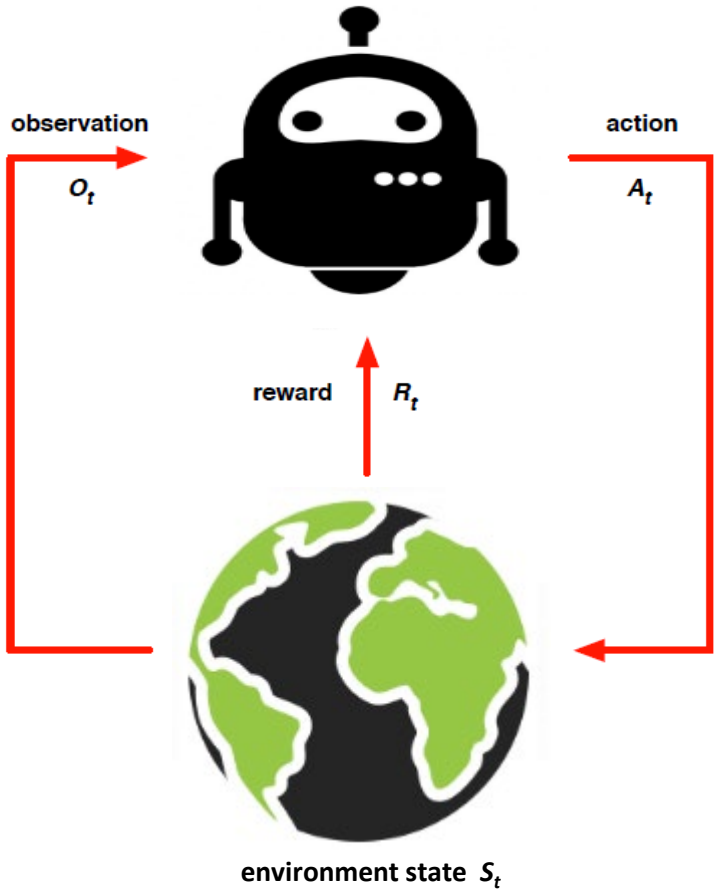
Dynamics: transition probabilities of states

$$\Pr\left(S_{t+1} = s' \middle| S_t = s, A_t = a\right)$$

where the MDP assumption implies

$$\Pr\left(S_{t+1} = s' \middle| S_t = s, A_t = a, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}\right) = \Pr\left(S_{t+1} = s' \middle| S_t = s, A_t = a\right)$$

Markov Decision Process (MDP)



Total reward G_t over an episod:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots.$$

Agent's policy:

$$\pi(a|s) = \Pr(\text{take action } a | \text{in state } s)$$

One of our goals will be to find the policy with the highest expected total reward:

$$\pi^*(a|s) = \arg\max_{\pi} E_{\pi}\{G_t\}$$

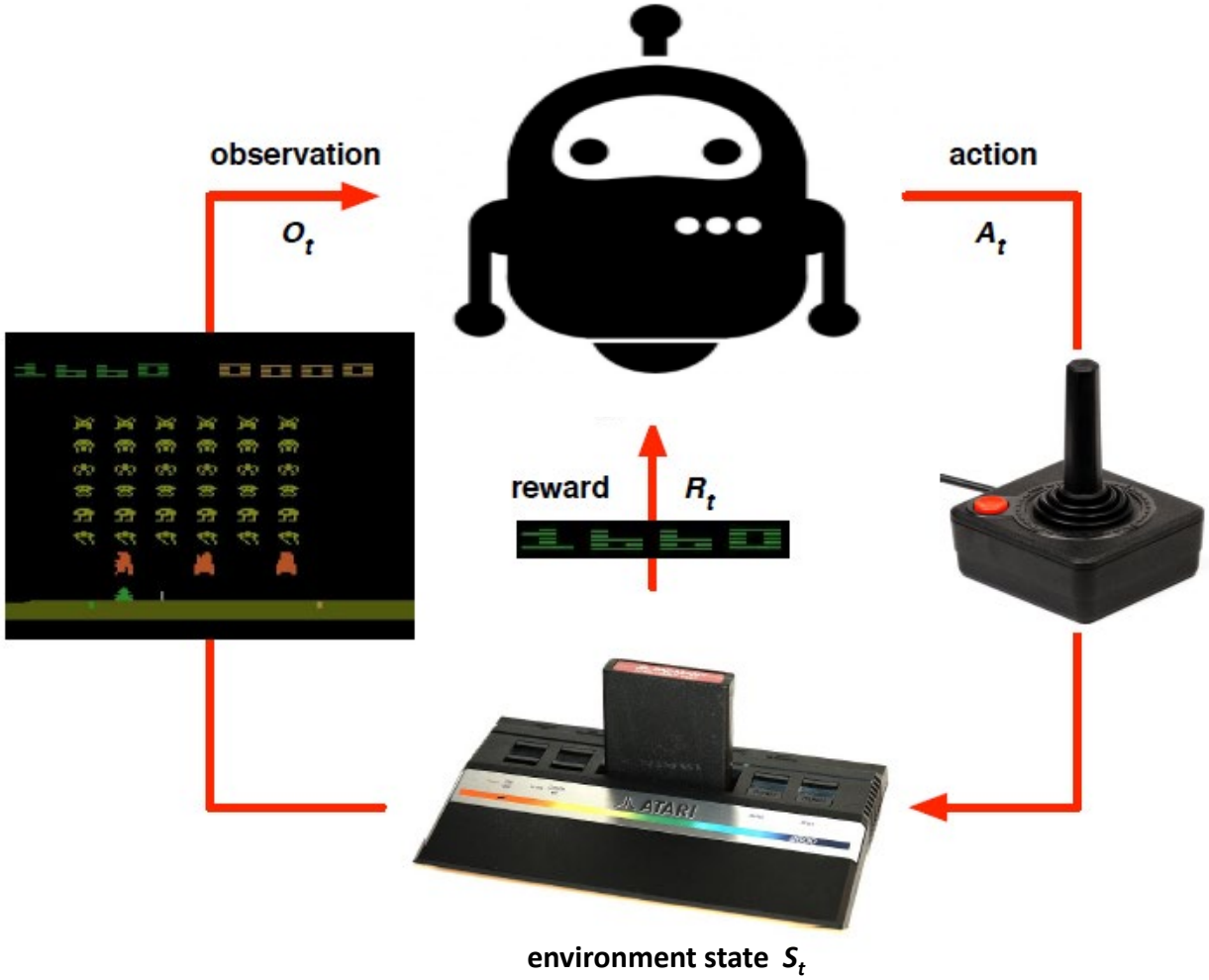
Maximisation of reward

- The **value** of a state *s*, is defined as the expected cumulative reward:

$$v_\pi(s) = E_\pi\left\{G_t \middle| S_t = s\right\} = E_\pi\left\{R_{t+1} + R_{t+2} + R_{t+3} + ... \middle| S_t = s\right\}$$

- Goal is to maximize the value, by picking most suitable actions.
- Values define desirability of a state and actions.
- It may be better to sacrifice immediate reward to gain long-term reward.

For the Atari gamer, identify all elements of an MDP…

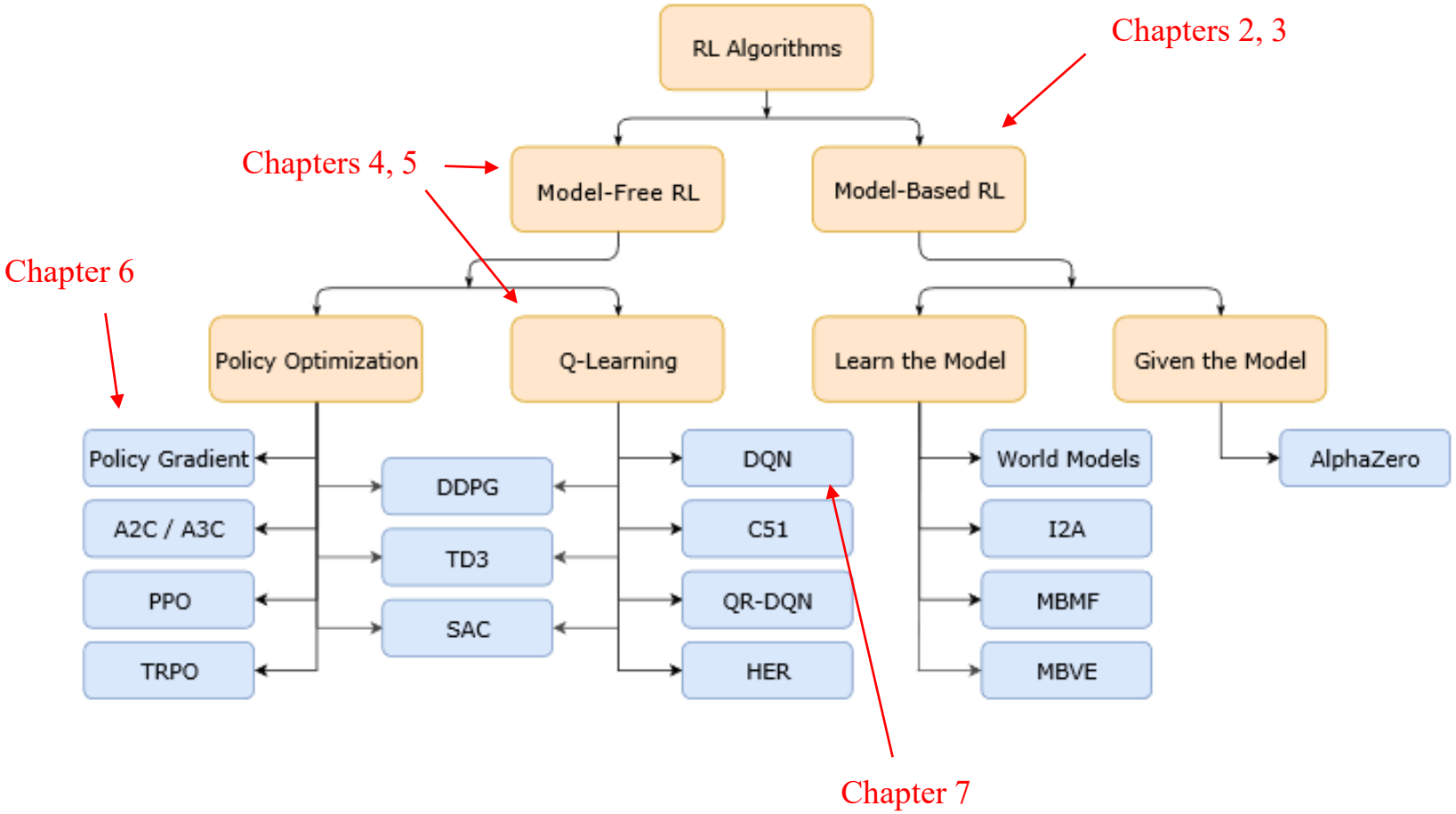


A thermostat controller, identify all elements of an MDP…

Key concepts in short…

- Two paradigms:
  - Model-based: environment behaviour is known
  - Model-free RL: there is no prior knowledge of the world
- Trade-off between exploration and exploitation:
  - Trial-and-error search
  - Optimize reward
- In RL there is no supervisor, only a reward signal.
- Feedback is possibly delayed, not instantaneous.
- Decisions are taken sequentially (time really matters).
- Agent's actions affect the environment, and hence the feedback it receives.

A taxonomy of RL



From [OpenAI Spinning Up]