

# Reinforcement Learning

---

## Lecture 1: Introduction to RL

Christopher Mutschler

# Opening Remarks

## Class Logistics

---

- We (will start with) use StudOn for main communication (forum + messages, announcements)
- If you have any questions, you can also write to



Chris

christopher.mutschler@iis.fraunhofer.de



Alex

alexander.mattick@iis.fraunhofer.de



Nico

nico.meyer@iis.fraunhofer.de

- If we will ever use a password somewhere, it will be **FAU\_RL\_2025**

# Opening Remarks

## Syllabus

Earn bonus points for the exam!

Week	Lecture		Exercises		
1.	14.04.	Intro to RL, MDPs	22.04.	no exercises	
2.	01.05.	public holiday	29.04.	MDPs	TA: Nico
3.	08.05.	Dynamic Programming	06.05.	<b>T.B.D.</b>	
4.	15.05.	Model-free Prediction	13.05.	DP	TA: Alex
5.	22.05.	Model-free Control	20.05.	OpenAI Gymnasium + PyTorch, TD-Learning	TA: Nico
6.	29.05.	public holiday	27.05.	TD-Control	TA: Nico
7.	05.06.	Value Function Approximation	03.06.	<b>Intermediate Exam</b>	
8.	12.06.	Policy-based RL #1	10.06.	no lectures	
9.	19.06.	public holiday	17.06.	DQN	TA: Nico
10.	26.06.	Policy-based RL #2	24.06.	VPG	TA: Alex
11.	03.07.	Exploration-Exploitation, Regret, Bandits	01.07.	A2C	TA: Nico
12.	10.07.	Exploration in Deep RL, Intrinsic Motivation	08.07.	Multi-Armed Bandits	TA: Alex
13.	17.07.	Model-based RL with Discrete Actions	15.07.	RND/ICM	TA: Alex
14.	24.07.	Guest Lecture (T.B.D.) Course Wrap-Up, Evaluation Results	22.07.	MCTS	TA: Alex

# Opening Remarks

## Syllabus

---

### **Tips & Tricks: How to survive this class**

- Play with the Jupyter notebooks (if we provide them) and exercise code
- Attend the classes, follow the content, and ask questions
- Attend the exercises and implement them
  
- ...one more thing
  - This is not a class to obtain 5 ECTS "as easy as it might be"

# Opening Remarks

## Exam

---

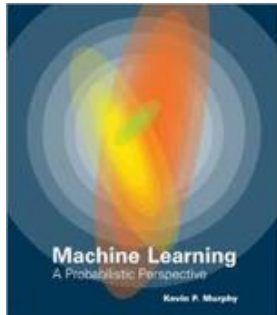
- Currently, the plan is to have a **written** exam.
- Final scheduling and logistics of the exam will be done around June/July
  - Exam will (most likely) take place in the first examination period! (planned: **28./29.07.2025**)
- The exams will cover topics from both **lectures** and **exercises**
  - Exams will focus on **basic understanding** of concepts
  - Exams will have **theoretical parts** (but we will not include proofs)
  - Exams will focus on **practical aspects** (i.e., implementation w.r.t. exercise content)
- Exam is in English (in the unlikely case of oral exams we can also do it in German)

# Opening Remarks

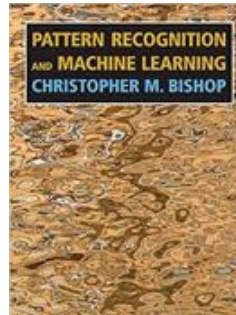
## Prerequisites

- What pre-requisites do we expect?
  - Analysis/Calculus
  - Multivariate Statistics
  - Machine Learning, Deep Learning
  - Python (to get used to the exercises)
- How can you get them?
  - You attended the recommended basic lectures: MLTS, Pattern Recognition, Deep Learning
  - Or/And dive into one of those books (better today than tomorrow):

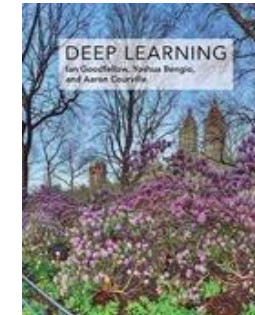
**Do not be so casual about it!**



*Kevin Murphy:  
Machine learning; a  
probabilistic  
perspective.*



*Christopher Bishop:  
Pattern Recognition  
and Machine Learning*



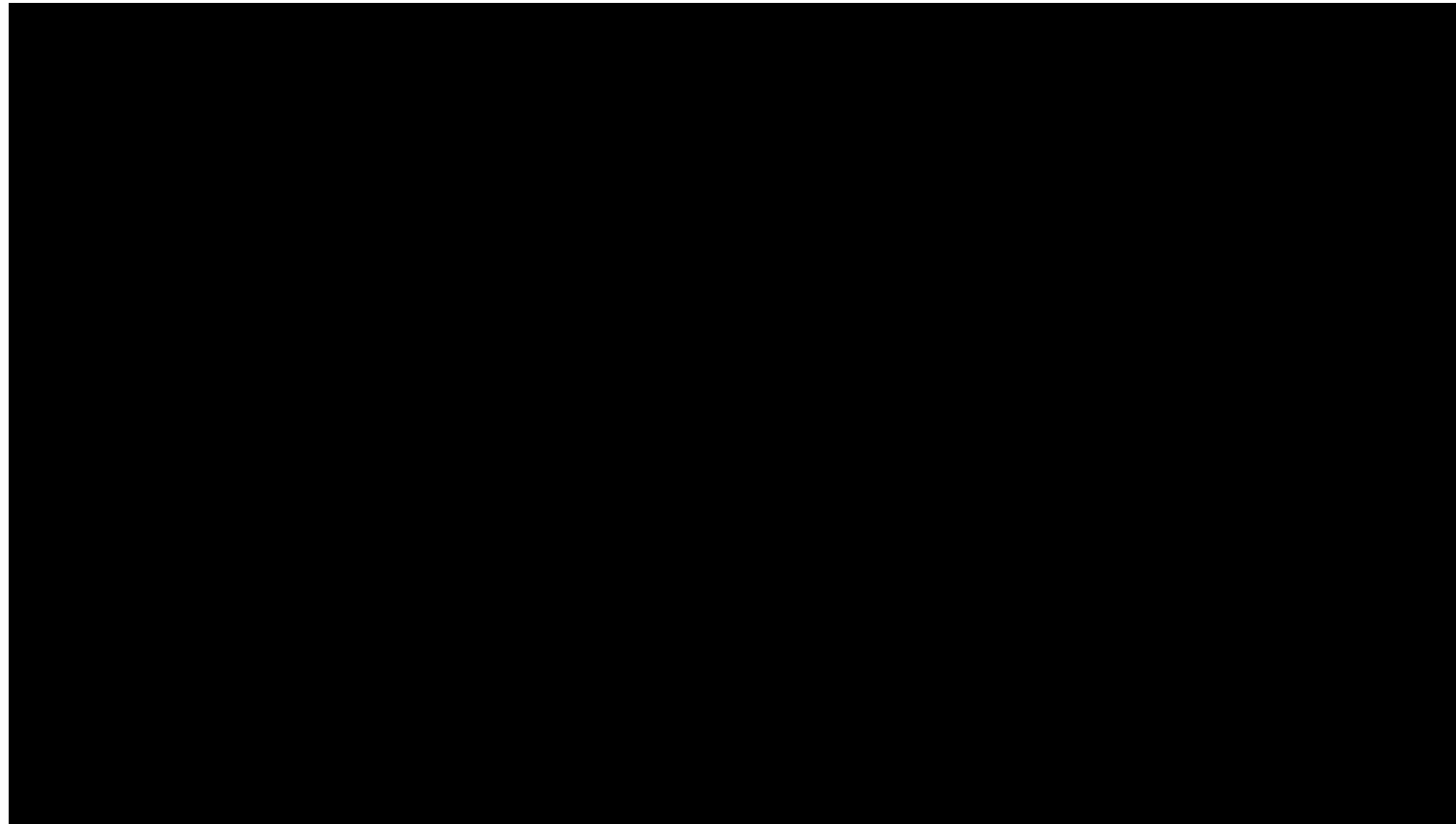
*Ian Goodfellow and Yoshua  
Bengio and Aaron Courville:  
Deep Learning*

- You will find RL literature on some later slides today



# Introduction to Reinforcement Learning

## Finding multi-agent soccer strategies with RL



<https://www.youtube.com/watch?v=F8DcgFDT9sc>

see also: <https://github.com/google-research/football>

# Introduction to Reinforcement Learning

## Controlling robots with RL



<https://www.youtube.com/watch?v=0JL04JJjocc>



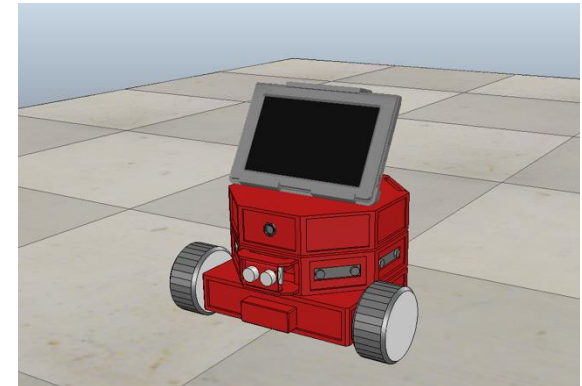
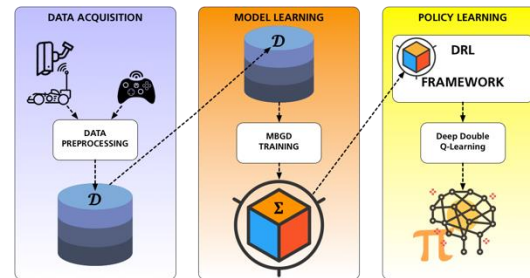
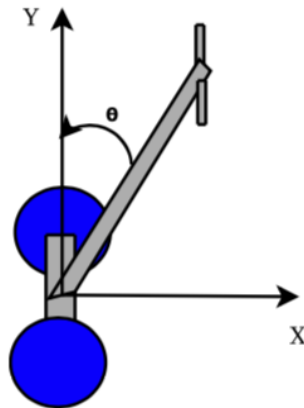
[https://www.youtube.com/watch?v=W\\_gxLKsSsIE](https://www.youtube.com/watch?v=W_gxLKsSsIE)

# Introduction to Reinforcement Learning

## Controlling robots with RL



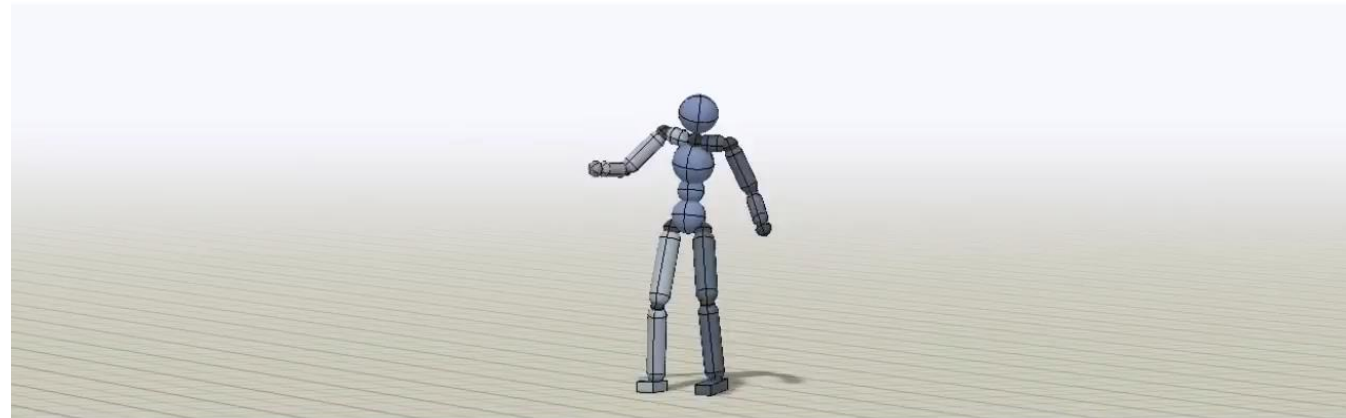
Deep Reinforcement Learning for  
On-line Collision-free Trajectory Planning  
in Dynamic Environments  
L. Butyrev, G. Kontes, T. Edelhäußer, C. Mutschler  
Submitted to ICRA 2019



# Introduction to Reinforcement Learning

## Advanced robot control in simulation

### DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills



Xue Bin Peng<sup>1</sup>, Pieter Abbeel<sup>1</sup>, Sergey Levine<sup>1</sup>, Michiel van de Panne<sup>2</sup>

<sup>1</sup> University of California  
Berkeley



<sup>2</sup> University of British  
Columbia



<https://www.youtube.com/watch?v=vppFvq2quQ0>

# Introduction to Reinforcement Learning

## Advanced robot control in reality

---



<https://www.youtube.com/watch?v=x4O8pojMF0w>


# Introduction to Reinforcement Learning

## Advanced robot control in reality

**Deep Drone Racing:  
Learning Agile Flight in Dynamic Environments**

Elia Kaufmann\*, Antonio Loquercio\*, Rene Ranftl,  
Alexey Dosovitskiy, Vladlen Koltun, Davide Scaramuzza

 University of Zurich  
Department of Neuroinformatics

 ETH zürich

 University of Zurich  
Department of Informatics



\* contributed equally

<https://www.youtube.com/watch?v=8RILnqPxo1s>



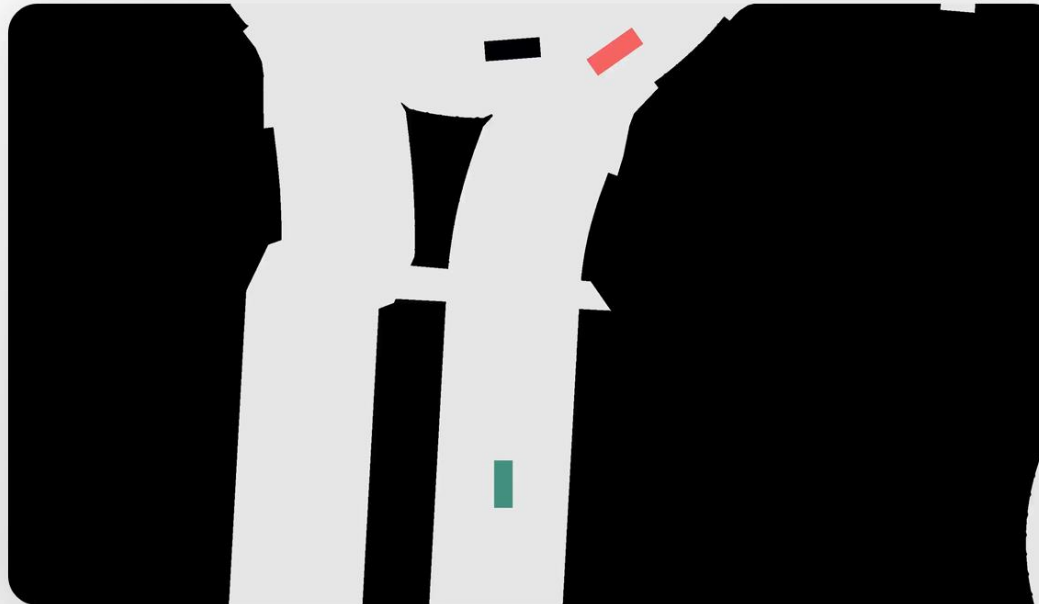
[https://www.youtube.com/watch?v=-e1\\_QhJ1EhQ](https://www.youtube.com/watch?v=-e1_QhJ1EhQ)

Not RL 😊

# Introduction to Reinforcement Learning

## Driving cars with RL

### AI Driver at Work Explainable Risk-Utility Tradeoff for autonomous Vehicles

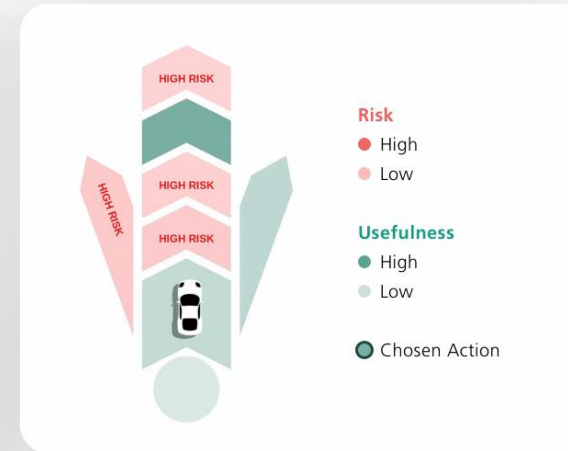


■ Agent vehicle

— Agent's planned route

■ Car is **highly relevant** for agent's risk assessment

■ Car is **less relevant** for agent's risk assessment



# Introduction to Reinforcement Learning

## Path planning/navigation



<https://www.youtube.com/watch?v=v5HjPSAK7k>



<https://www.youtube.com/watch?v=H7Ym3DMSGms>

# Introduction to Reinforcement Learning

## Practical Implementations



 Google DeepMind

### Energy efficiency at Google's data centers

- Every 5 minutes, AI draws snapshots of data center cooling system through thousands of sensors
- Information fed into deep neural network, → defines optimal action to reduce energy usage while keeping data center reliable
- Actions verified by second system and then implemented
- Significant energy and cost savings achieved

Source: <https://deepmind.google/discover/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control/>



 InstaDeep™

### Optimizing Rail Network with RL

- In case of unexpected events, trains need to be redirected in real time
- Ensure optimal capacity planning and traffic management in real time
- RL trained with real data and then further improved with millions of simulations
- Through optimized capacity planning and traffic management  
→ higher train frequencies  
→ delays can be better avoided

Source: <https://www.instadeep.com/2023/04/instadeep-explores-the-benefits-of-deep-reinforcement-learning-technology-in-transportation-at-interchange-event/>



### Ballons steering in stratosphere

- Ballons that autonomously operate in stratosphere for months serving various purposes
- Complex steering involves considering cues like wind speed, visibility, solar elevation, and (imperfect) weather forecasts
- RL trained with millions of simulated flight hours to make optimal real-time decisions
- RL surpasses previous algorithms and withstands natural diversity

Source: <https://blog.x.company/drifts-efficiently-through-the-stratosphere-using-deep-reinforcement-learning-c38723ee2e90>

# Introduction to Reinforcement Learning

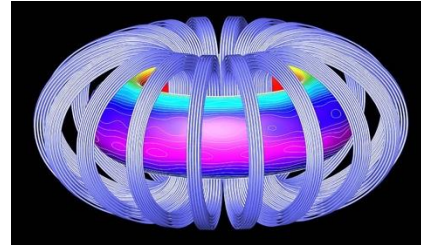
## Practical Implementations



### RL to optimize the manufacturing flow of semiconductors at Infineon

- RL-driven optimization of semiconductor manufacturing flow
- RL agent automates order dispatching in wafer fabrication
- Wafers automatically directed specific equipment for processing, considering limited capacity and storage space
- Ensures smooth flow with minimal delays
- Outperformed traditional heuristic methods

Source: <https://www.sciencedirect.com/science/article/pii/S0007850618300659?via%3Dihub>



### Optimizing nuclear fusion with Deep RL at Swiss Plasma Center in Lausanne

- Utilization of RL for precise control of plasma (heated hydrogen) in tokamak for nuclear fusion
- Plasma is confined within a vacuum room with magnetic coils
- Plasma instability requires constant adjustment of magnetic coils & other conditions
- RL optimizes adjustments considering multiple conditions simultaneously
- Potential clean energy source

Source: <https://deepmind.google/discover/blog/accelerating-fusion-science-through-learned-plasma-control/>; <https://www.nature.com/articles/s41586-021-04301-9>



### RL to solve American Airlines inventory control & overbooking problems

- Utilization of RL to manage overbookings
- Considers multiple factors to maximize revenue and minimize bumping costs
- Ensures flights remain full despite cancellations
- RL agent surpasses traditional methods, yielding close to maximum profits per flight

Source: <https://arxiv.org/pdf/1902.06824.pdf>

# Introduction to Reinforcement Learning

## Practical Implementations



ZARA

### RL for inventory optimization and shortened lead times at Zara

- AI-driven system optimizes production and supply chain
- RL agent leverages sales data, customer feedback, and social media trends to predict demand
- RL-driven recommendations allow optimization of inventory levels due
- Shortened lead & delivery times enhance customer satisfaction by ensuring goods are available when needed

Source: <https://www.tokinomo.com/blog/artificial-intelligence-in-retail>



Inria

Université de Lille

### Optimizing crop management efficiency with open-source RL

- Farmers must balance multiple variables for optimal crop production, e.g., weather, fertilization, & soil conditions
- RL agent simultaneously considers various factors to provide recommendations for optimal crop management
- Facilitates reaching production objectives while optimizing resource utilization
- Open-source RL environment provided by Inria centre at the University of Lille

Source: <https://arxiv.org/pdf/2207.03270v1.pdf>



FAU  
FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG

SIEMENS

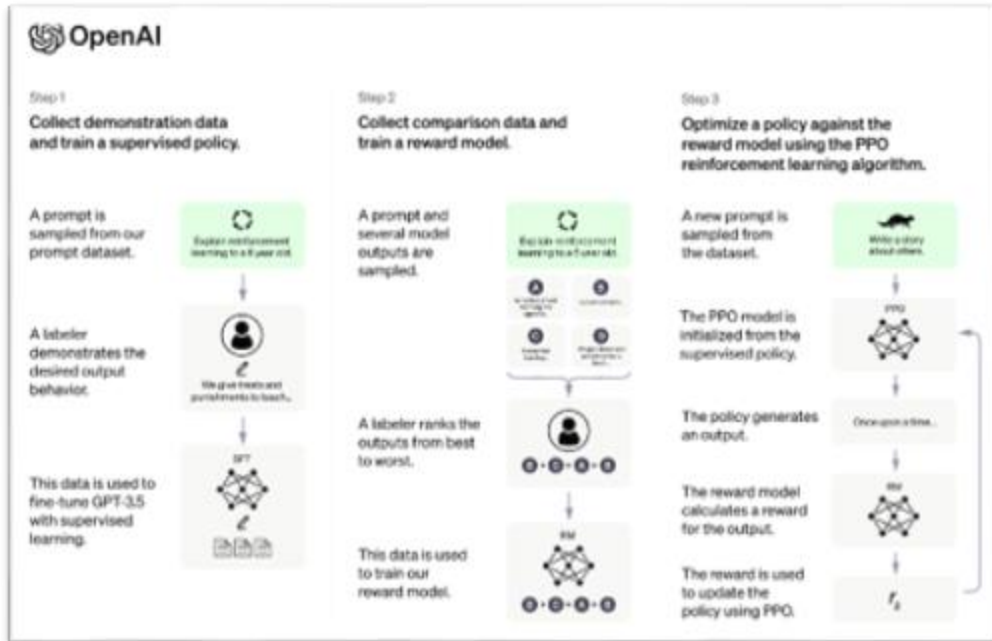
### Utilization of deep RL to optimize energy costs in a flexible production machine

- RL employed to ensure most effective control policy for production machines considering varying framework conditions (e.g., energy prices)
- Deep learning architecture forecasts load profiles of future manufacturing schedules from past production time series
- RL algorithm trained to optimize machine load and speed for long-term energy cost reduction

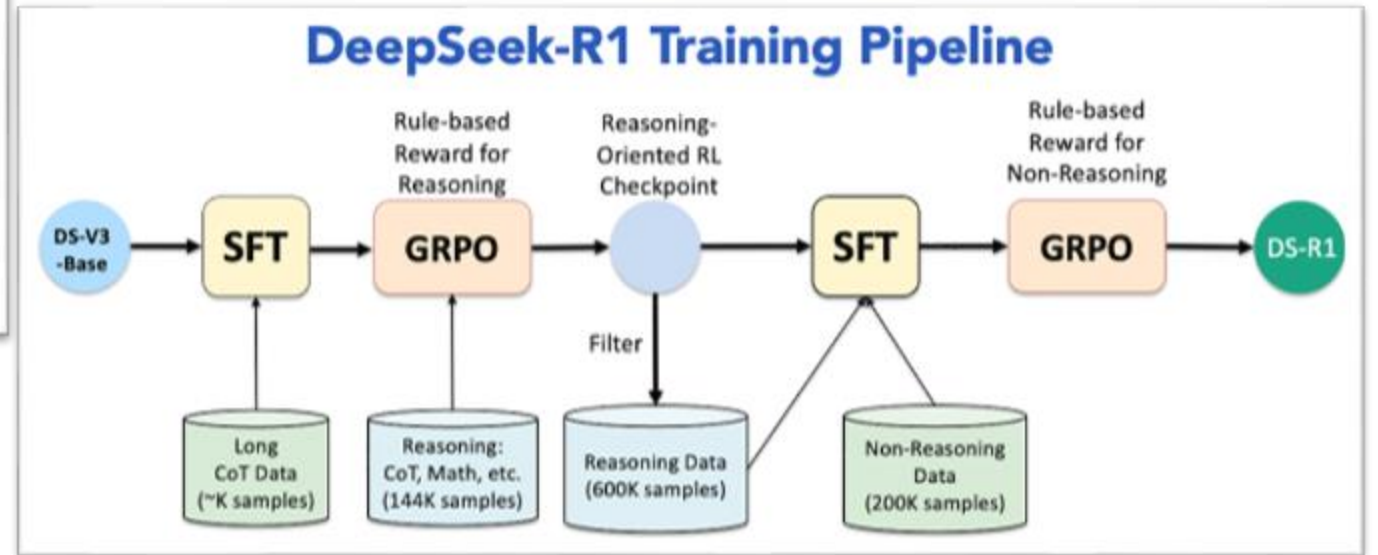
Source: <https://www.scientific.net/AMM.882.96.pdf>

# Introduction to Reinforcement Learning

## Practical Implementations



<https://openai.com/blog/chatgpt>



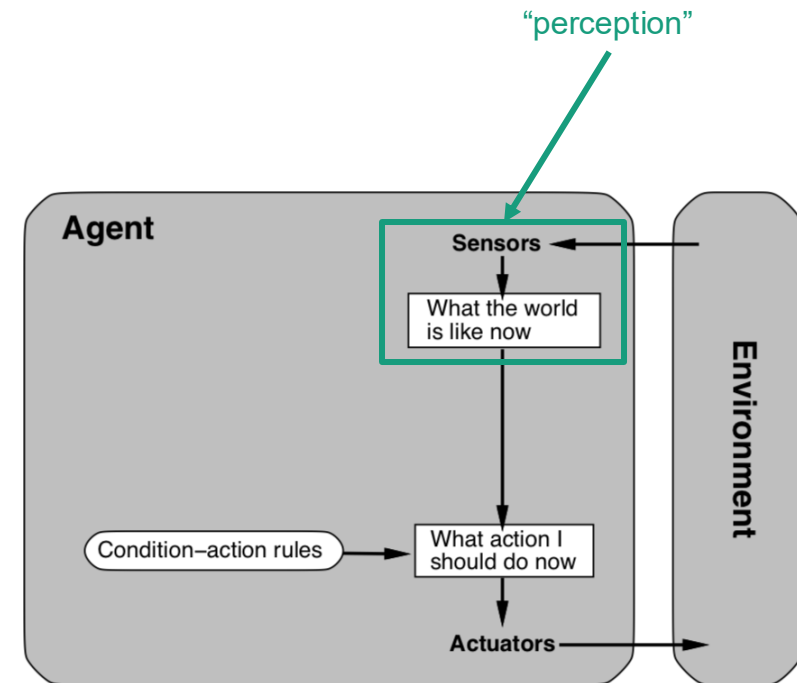
<https://medium.com/@lmpo/deepseek-r1-affordable-efficient-and-state-of-the-art-ai-reasoning-f293b0bd8d65>

# Introduction to Reinforcement Learning

## What are “autonomous systems”?

- **Autonomous Agent (Simple Reflex Agent)**
- An Autonomous Agent is **anything** that:
  - Perceives its environment via sensors
  - Acts on it with actuators
  - Operates without any interference (autonomously)

→ Percept & Act



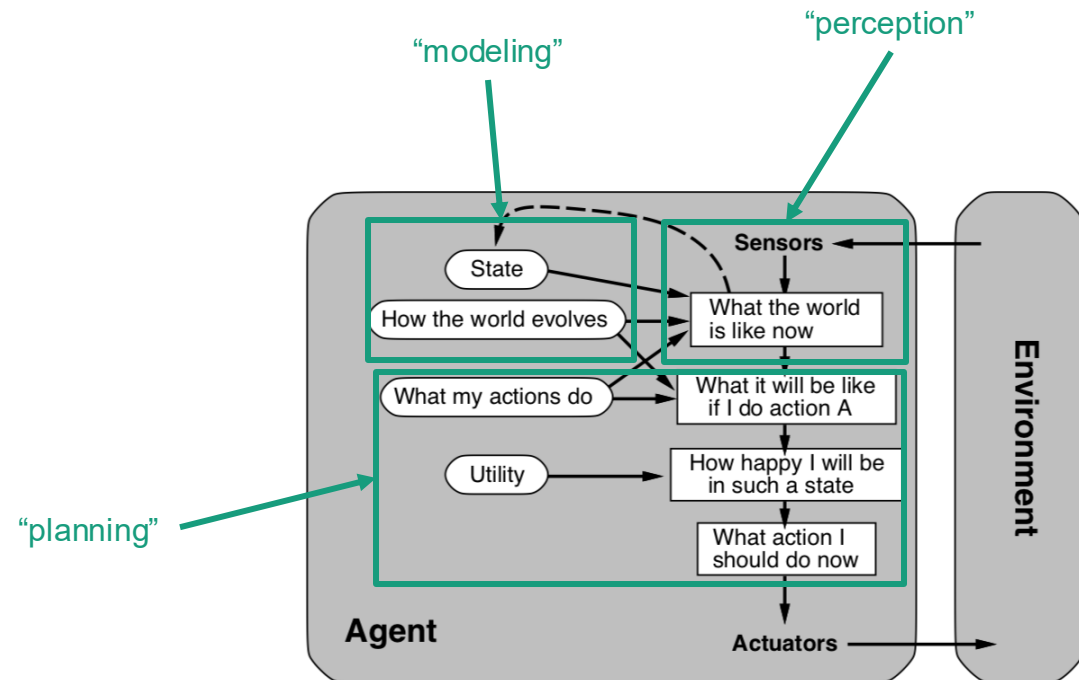
Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

# Introduction to Reinforcement Learning

## What are “autonomous systems”?

- **Intelligent (Utility-based) Agent**
- An intelligent agent is **anything** that:
  - Perceives its environment via sensors
  - Acts on it with actuators
  - Operates without any interference (autonomously)
  - Directs its activity towards achieving goals or maximizing a utility function

→ Percept & Plan to Control



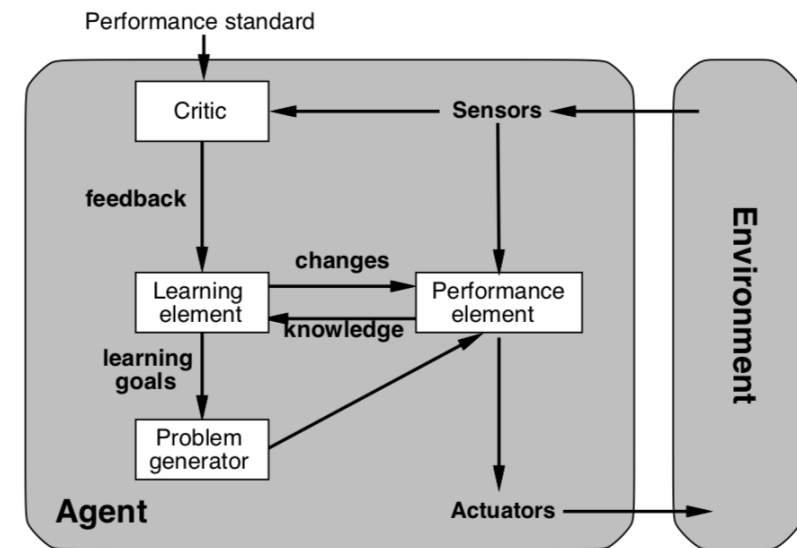
Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

# Introduction to Reinforcement Learning

## What are “autonomous systems”?

- **Learning Agent**
- A learning agent is **anything** that:
  - Perceives its environment via sensors
  - Acts on it with actuators
  - Operates without any interference (autonomously)
  - **Learns** how to better achieve goals or maximize a **utility function**

→ **Percept & Learn to Control**  
(not necessarily separate)



Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

# Introduction to Reinforcement Learning

## What are “autonomous systems”?

---

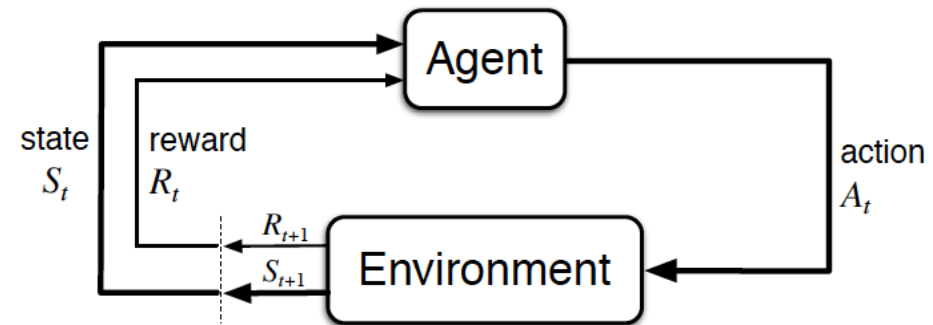
- Autonomous Cars
- Smart Homes/Buildings that adapt to occupants
- Intelligent traffic lights control
- Software trading agents
- Virtual assistants that manage appointments or answer emails automatically
- Recommender systems, e.g., for movies (Netflix), consumer products (Amazon), advertisements (Google), content (Facebook) or music (Spotify) recommendations
- Player Modelling and Content Generation in Computer Games

# Introduction to Reinforcement Learning

## The RL Paradigm (reward hypothesis)

- Do you agree with following statement?

*"All goals can be described by the maximization of expected cumulative **reward**."*

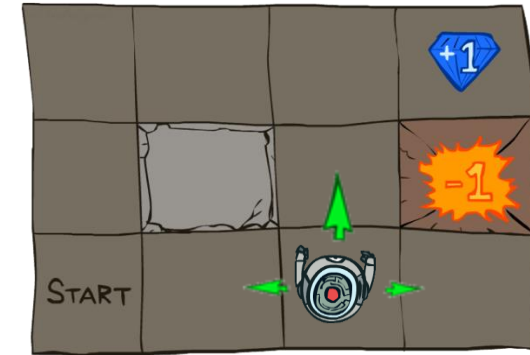


Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

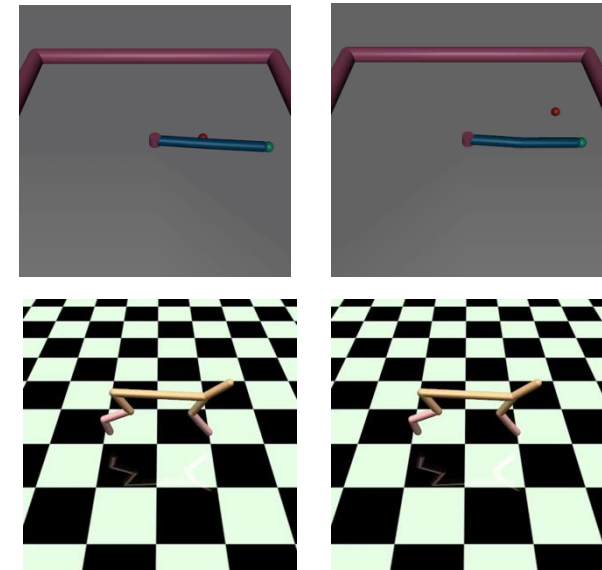
# Introduction to Reinforcement Learning

## Goals for different applications

- Control a robot in the Gridworld
  - Getting to the treasure
  - Falling into traps
- Play videogames
  - Increasing the score
  - Decreasing the score
- Fly stunt maneuvers in a helicopter
  - Following desired trajectory
  - Crashing
- Humanoid walk
  - Forward motion
  - Falling over

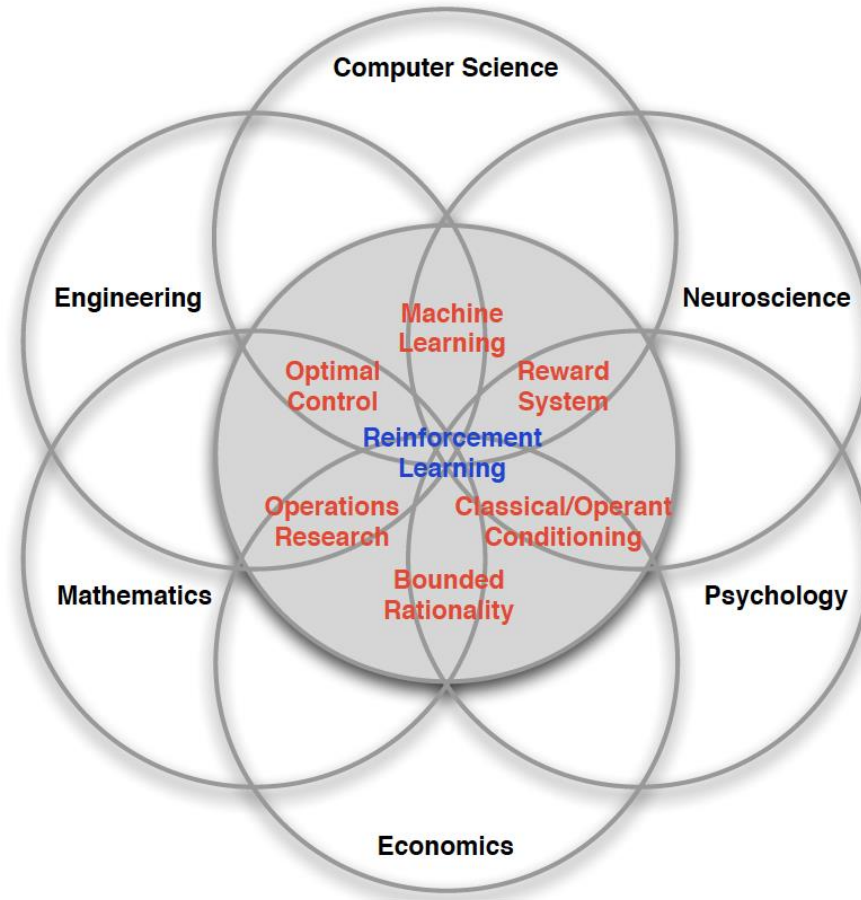


[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)



# Introduction to Reinforcement Learning

## RL vs the world: the many faces of Reinforcement Learning



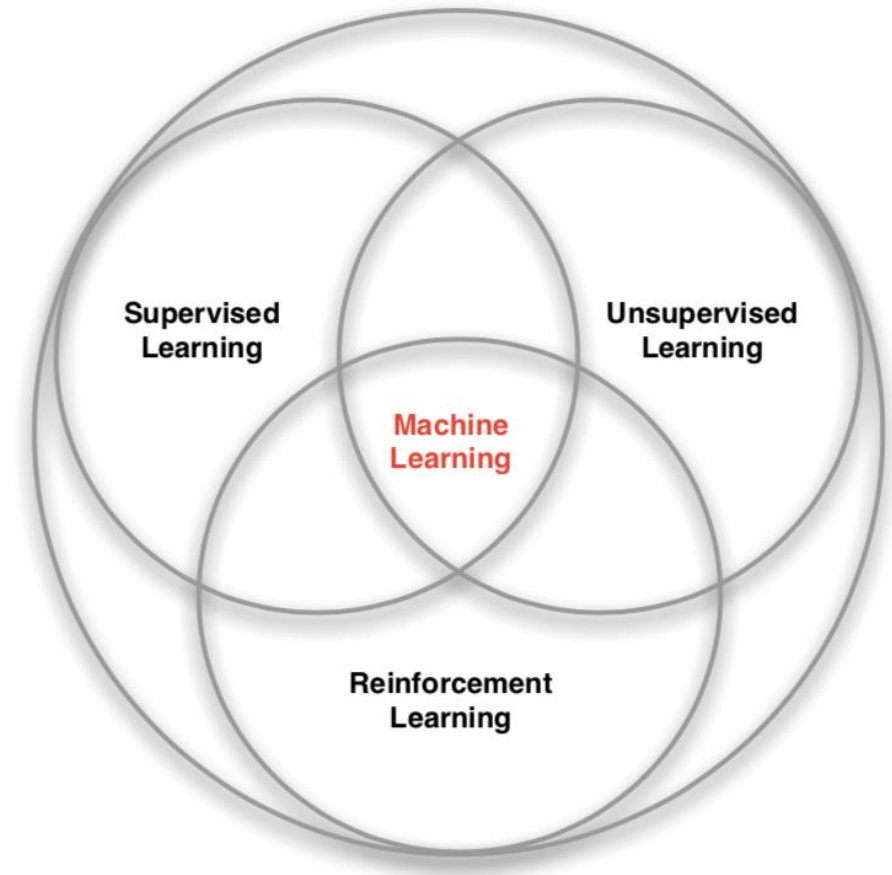
David Silver 2015

see also: <https://www.youtube.com/watch?v=-63ysqT5nu0>

# Introduction to Reinforcement Learning

## RL vs other ML branches

- No teacher/supervisor, only reward signals.
- Delayed feedback, not instantaneous (credit assignment problem).
- Learning by interaction between environment and agent over time.
- Agent's actions affect the environment:  
**Actions have consequences!!!**  
→ non i.i.d.!
- Active Learning process: the actions that the agent takes affect the subsequent data the agent receives

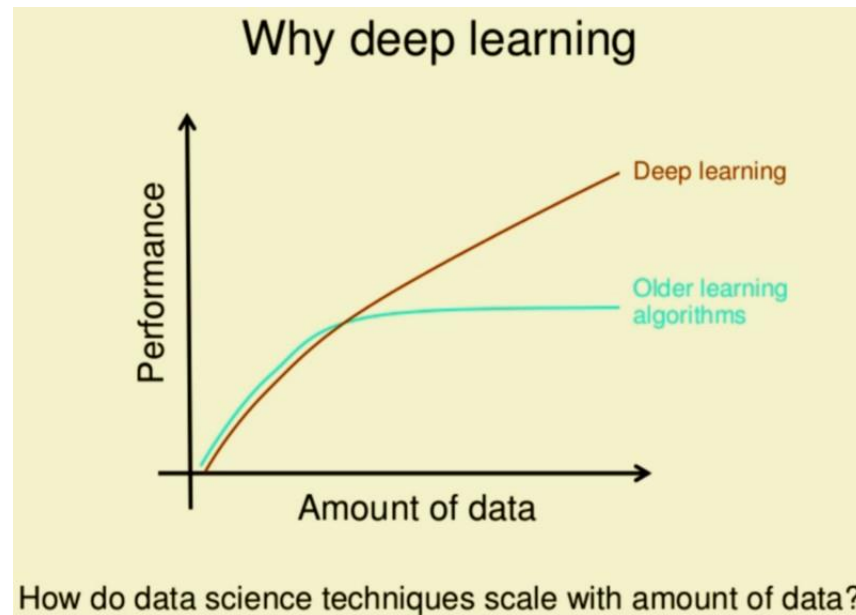


see also: <https://www.youtube.com/watch?v=-63ysqT5nu0>

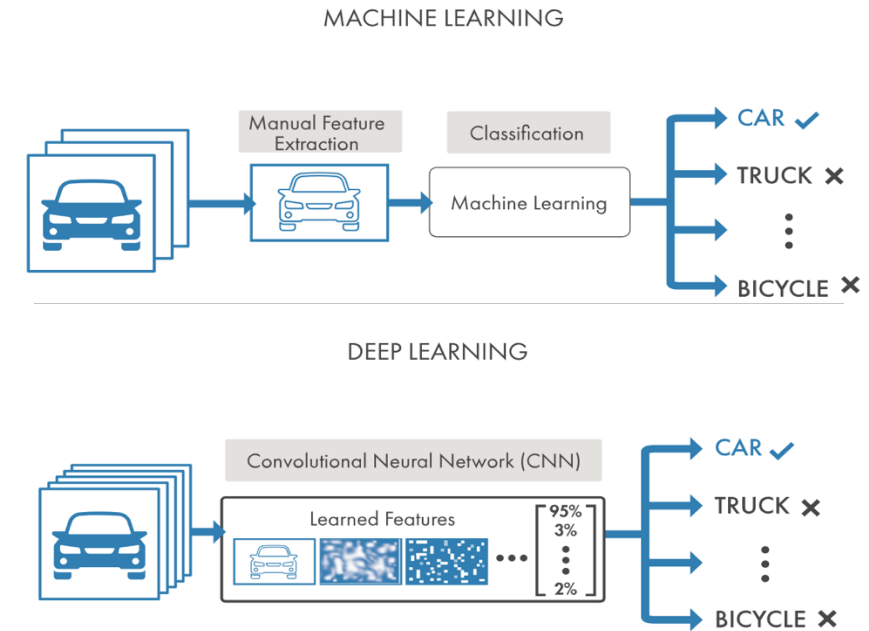
# Introduction to Reinforcement Learning

## Why RL now?

- Taking advantage of advances in:
  - **Deep Learning Algorithms (DL)**



<https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063>

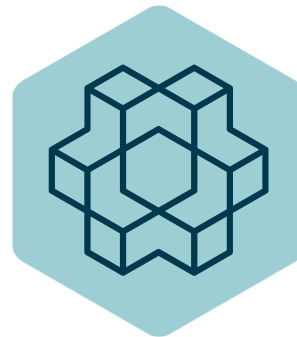


<https://www.mathworks.com/discovery/deep-learning.html>

# Introduction to Reinforcement Learning

## Why RL now?

- Taking advantage of advances in:
  - Deep Learning Algorithms (DL)
  - **Software for DL and RL**



# Introduction to Reinforcement Learning

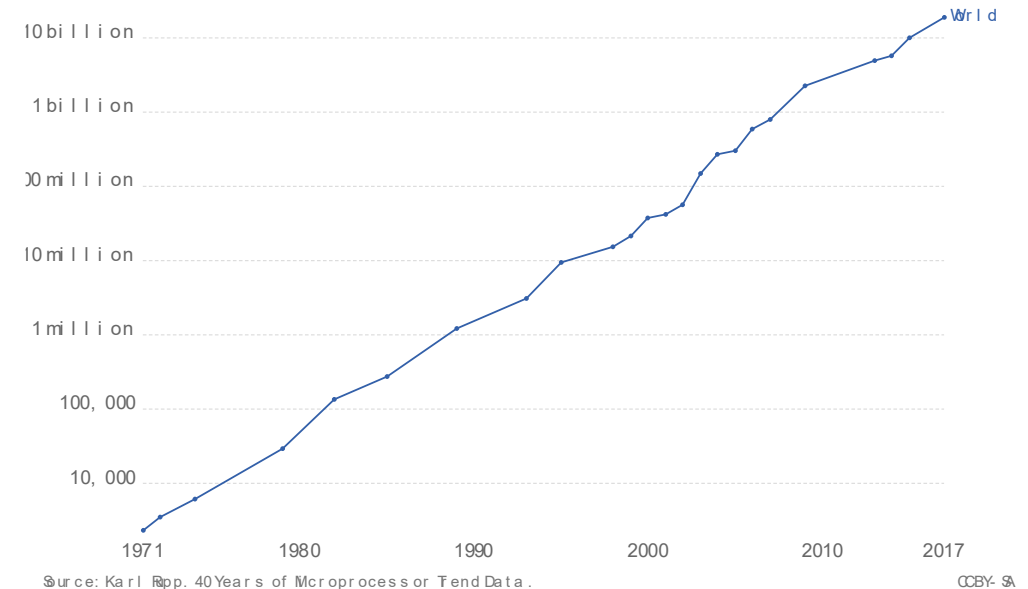
## Why RL now?

- Taking advantage of advances in:
  - Deep Learning Algorithms (DL)
  - Software for DL and RL
  - **Hardware (CPU & Memory)**

	OPENAI 1V1 BOT	OPENAI FIVE
<b>CPUs</b>	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
<b>GPUs</b>	256 K80 GPUs on Azure	256 P100 GPUs on GCP
<b>Experience collected</b>	~300 years per day	~180 years per day (~900 years per day counting each hero separately)

<https://blog.openai.com/openai-five/>

**More's law: Transistors per microprocessor**  
 Number of transistors which fit into a microprocessor. This relationship was famously first created to describe the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.

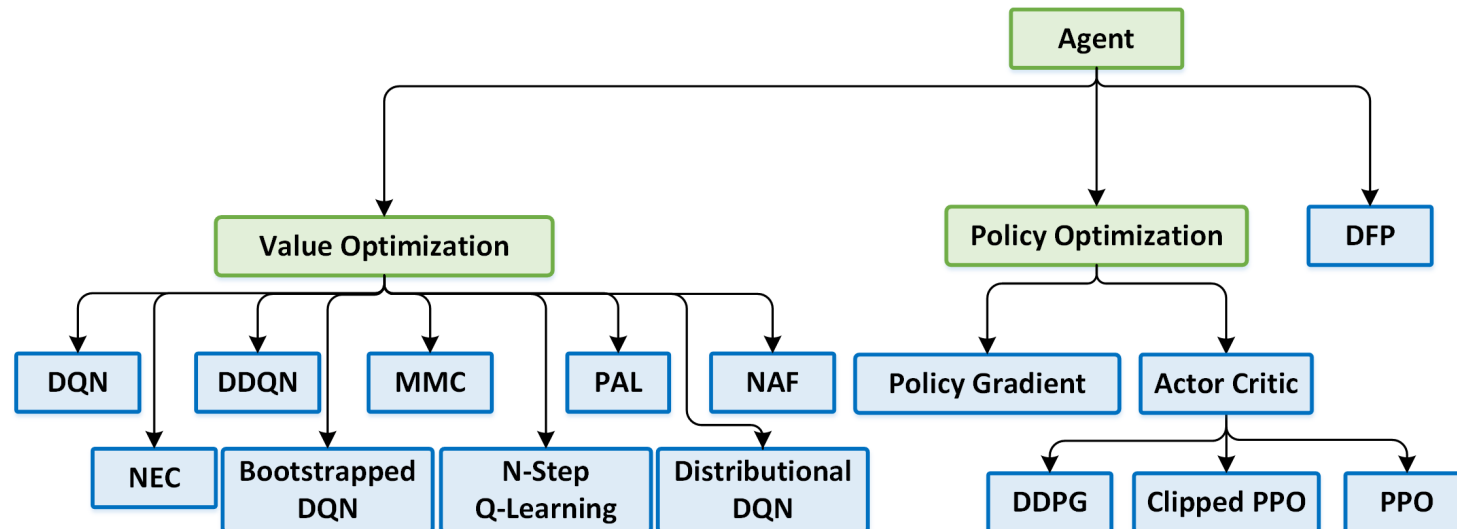


<https://ourworldindata.org/technological-progress>

# Introduction to Reinforcement Learning

## Why RL now?

- Taking advantage of advances in:
  - Deep Learning Algorithms (DL)
  - Software for DL and RL
  - Hardware (CPU & Memory)
  - **Deep RL**



<https://ai.intel.com/reinforcement-learning-coach-intel/>

# Introduction to Reinforcement Learning

## Why RL now?

---

- Taking advantage of advances in:
  - Deep Learning Algorithms (DL)
  - Software for DL and RL
  - Hardware (CPU & Memory)
  - Deep RL
  - **(Really good) Open Source Algorithm Implementations**



# Introduction to Reinforcement Learning

## Why RL now?

- Taking advantage of advances in:
  - Deep Learning Algorithms (DL)
  - Software for DL and RL
  - Hardware (CPU & Memory)
  - Deep RL
  - (Really good) Open Source Algorithm Implementations
  - **You!**



# Introduction to Reinforcement Learning Literature

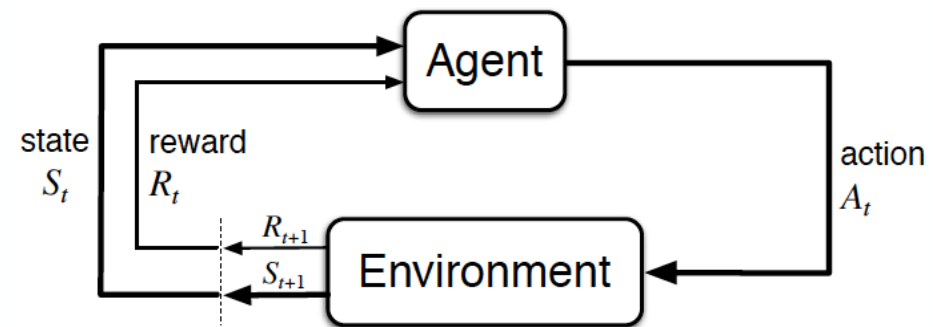


# Introduction to Reinforcement Learning

## The RL Paradigm (revisited)

- Do you agree with following statement?

*"All goals can be described by the maximization of expected cumulative **reward**."*

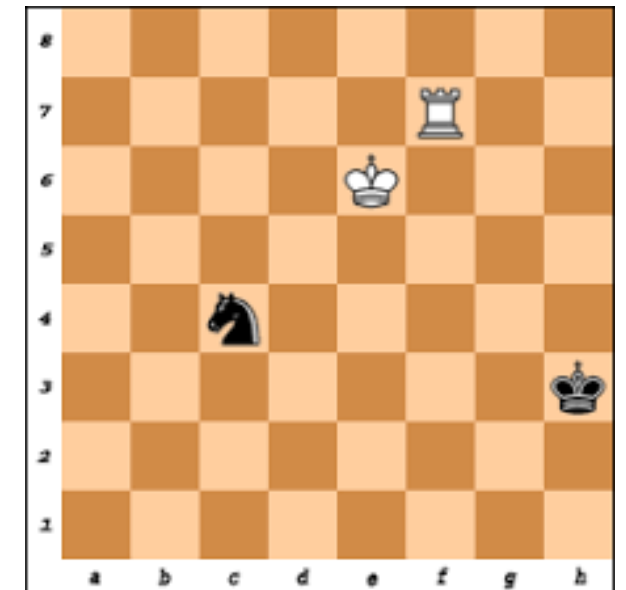


Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

# Introduction to Reinforcement Learning

## Challenges of sequential decision making

- **Goal: select actions to maximize total future reward**
- Actions may have long-term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
  - Financial investments
  - Refueling the helicopter
  - Game playing?

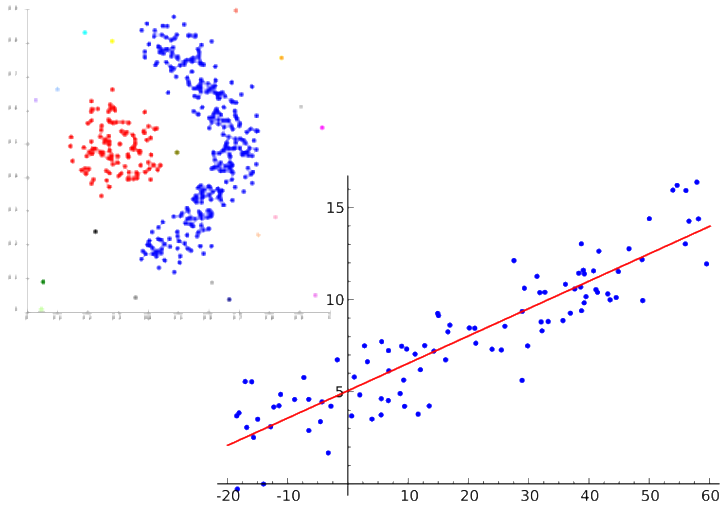


# Introduction to Reinforcement Learning

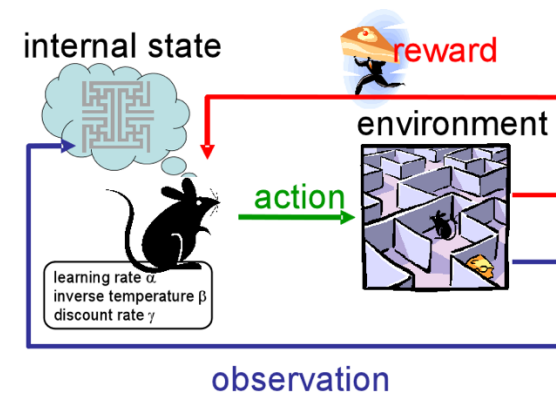
## Challenges of understanding/adopting RL

- Counter-Intuitive Visualization!!!

Supervised Learning



Reinforcement Learning

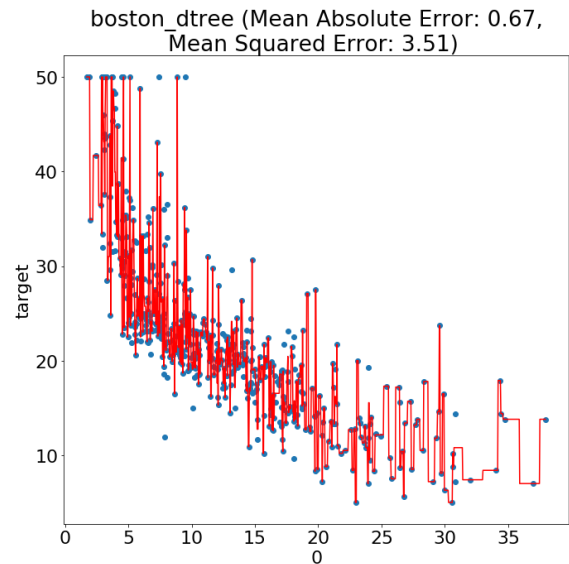


# Introduction to Reinforcement Learning

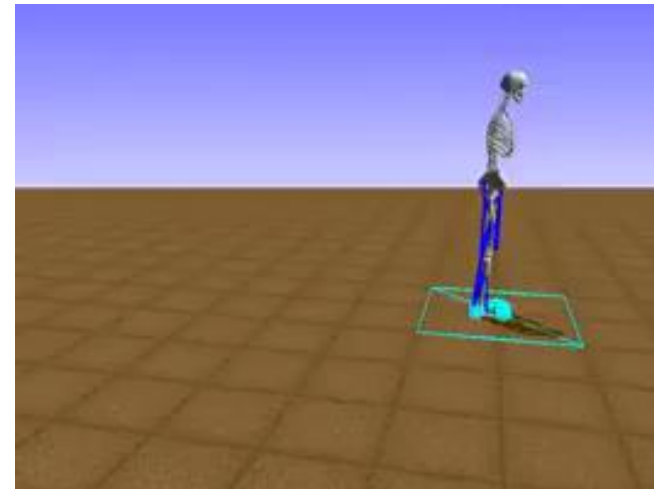
## Challenges of understanding/adopting RL

- Example: what went wrong here?

### Supervised Learning



### Reinforcement Learning



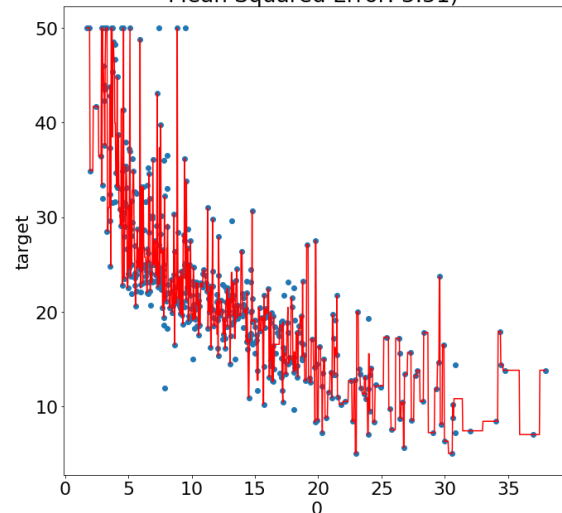
# Introduction to Reinforcement Learning

## Challenges of understanding/adopting RL

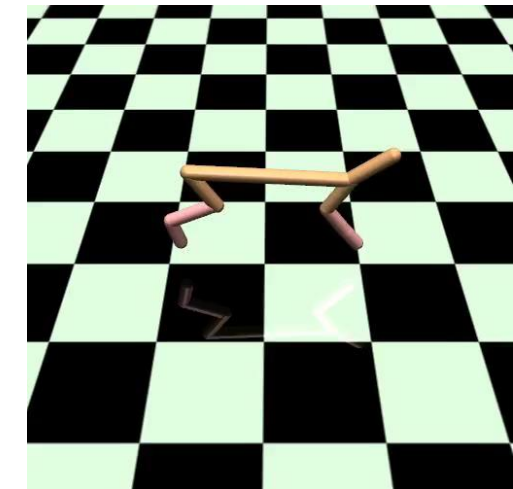
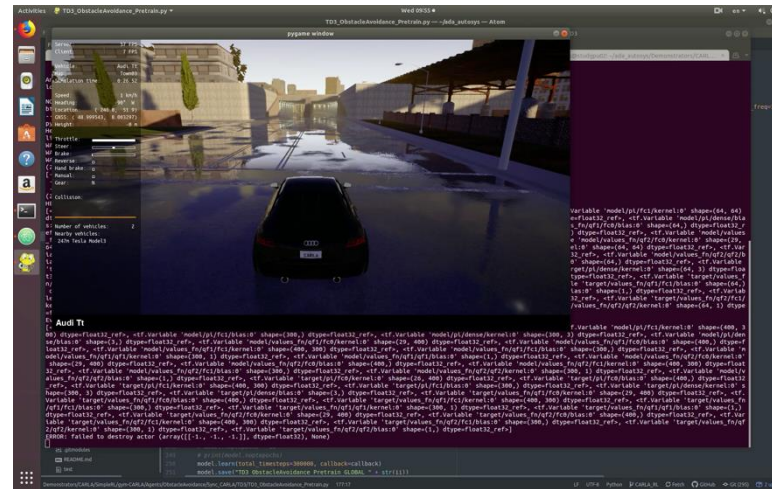
- Example: what went wrong here?

### Supervised Learning

boston\_dtree (Mean Absolute Error: 0.67, Mean Squared Error: 3.51)



### Reinforcement Learning

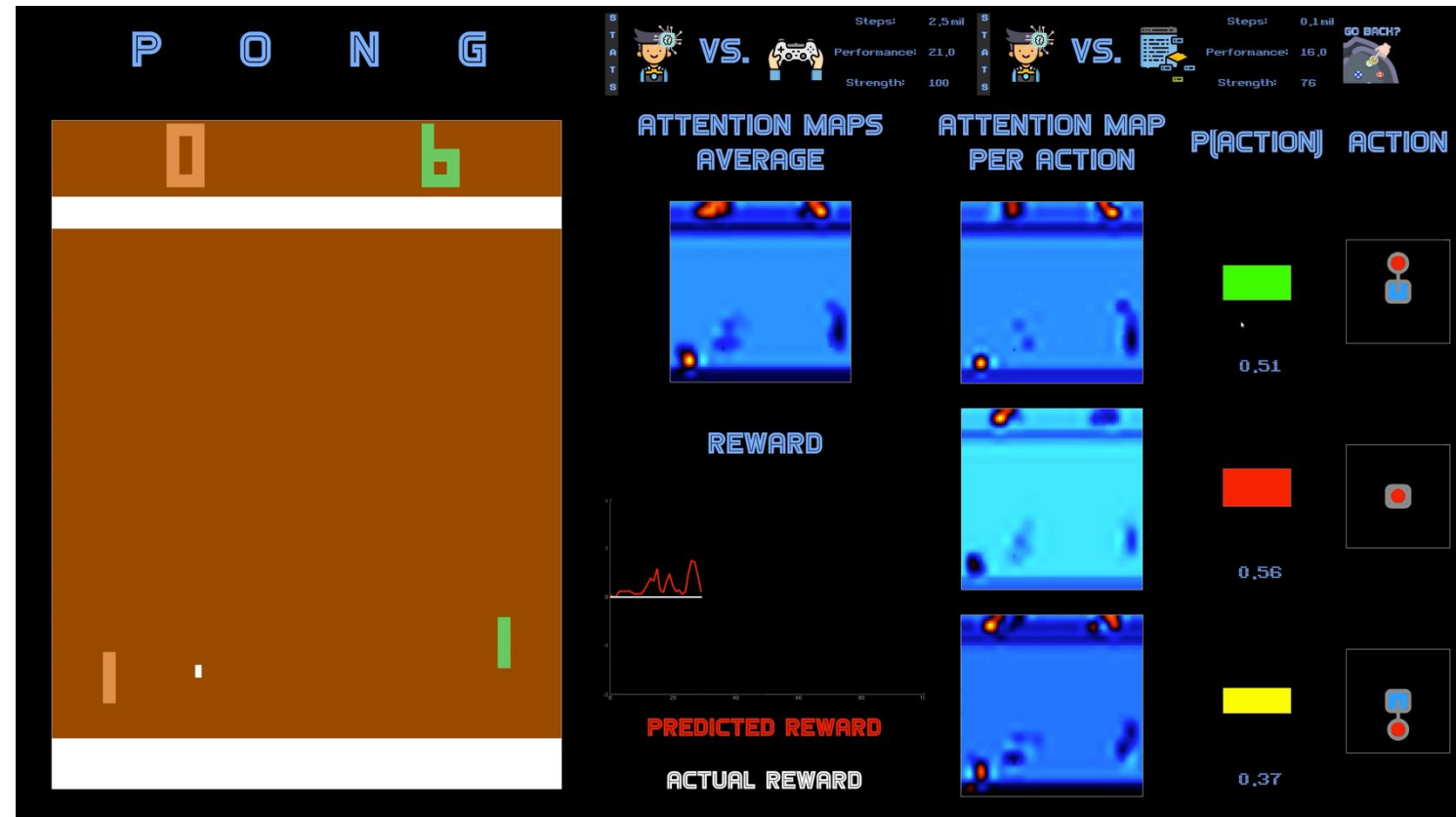


[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)

# Introduction to Reinforcement Learning

## Challenges of understanding/adopting RL

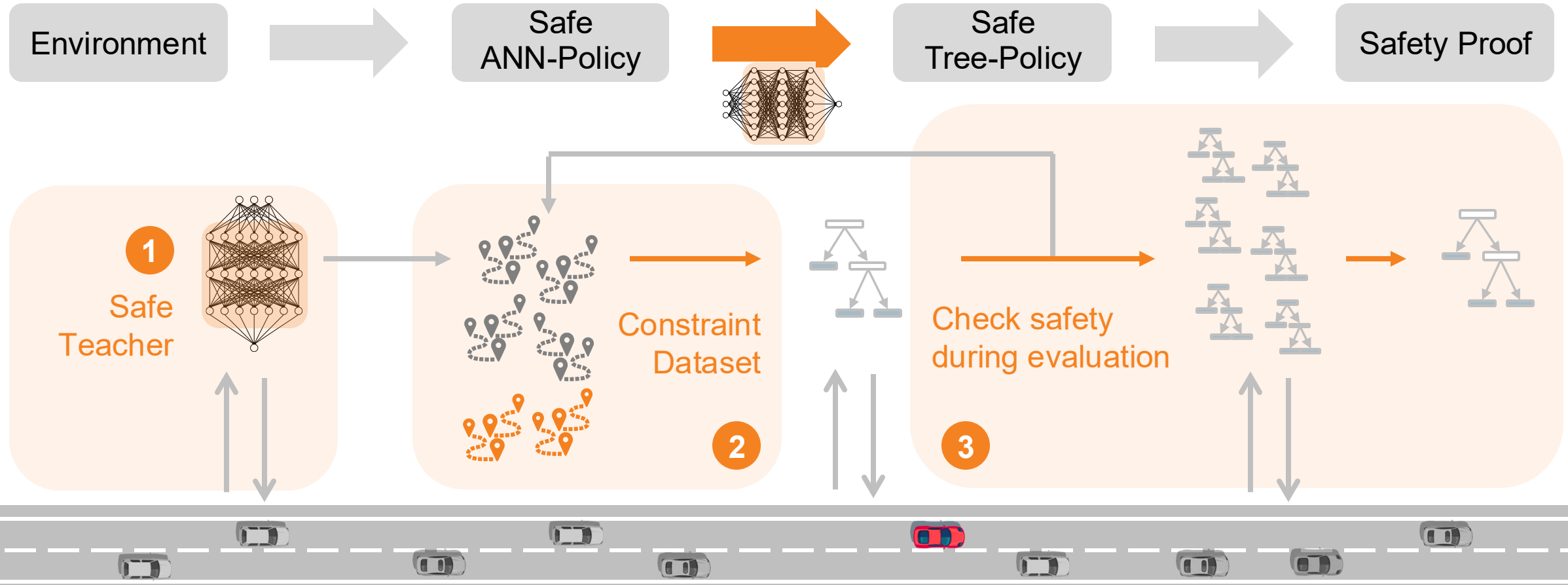
- Idea: Saliency Maps



# Introduction to Reinforcement Learning

## Challenges of understanding/adopting RL

- Idea: explainable decision rules



# Introduction to Reinforcement Learning

## Challenges of understanding/adopting RL

---

- Simple algorithms don't scale!!!
  - k-means → time-series clustering
  - Linear/polynomial regression → house/car pricing prediction
  - Tabular Q-Learning/SARSA → very specialized applications

# Introduction to Reinforcement Learning

## Myth vs. Reality

---

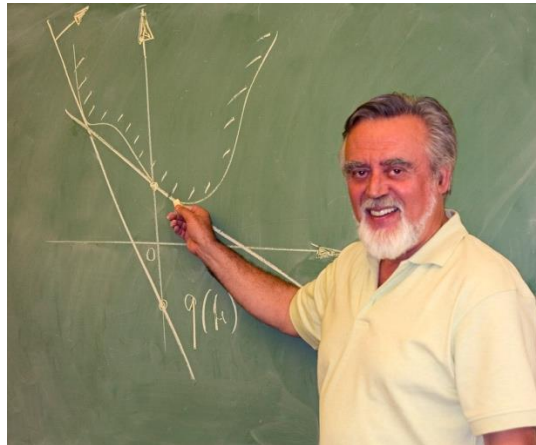
1. AI is RL  
→ NO! Many AI methods exist
2. RL can solve only games  
→ NO! We will see several examples
3. RL is just "fancy" search  
→ NO! We will compare to fancy search methods and see this
4. (Deep) RL can solve any problem, without any domain knowledge  
→ NO!

# Introduction to Reinforcement Learning

## Myth vs. Reality

- Deep RL can solve anything vs Deep RL does not work  
(see <https://www.alexirpan.com/2018/02/14/rl-hard.html>)

**NO and NO!**

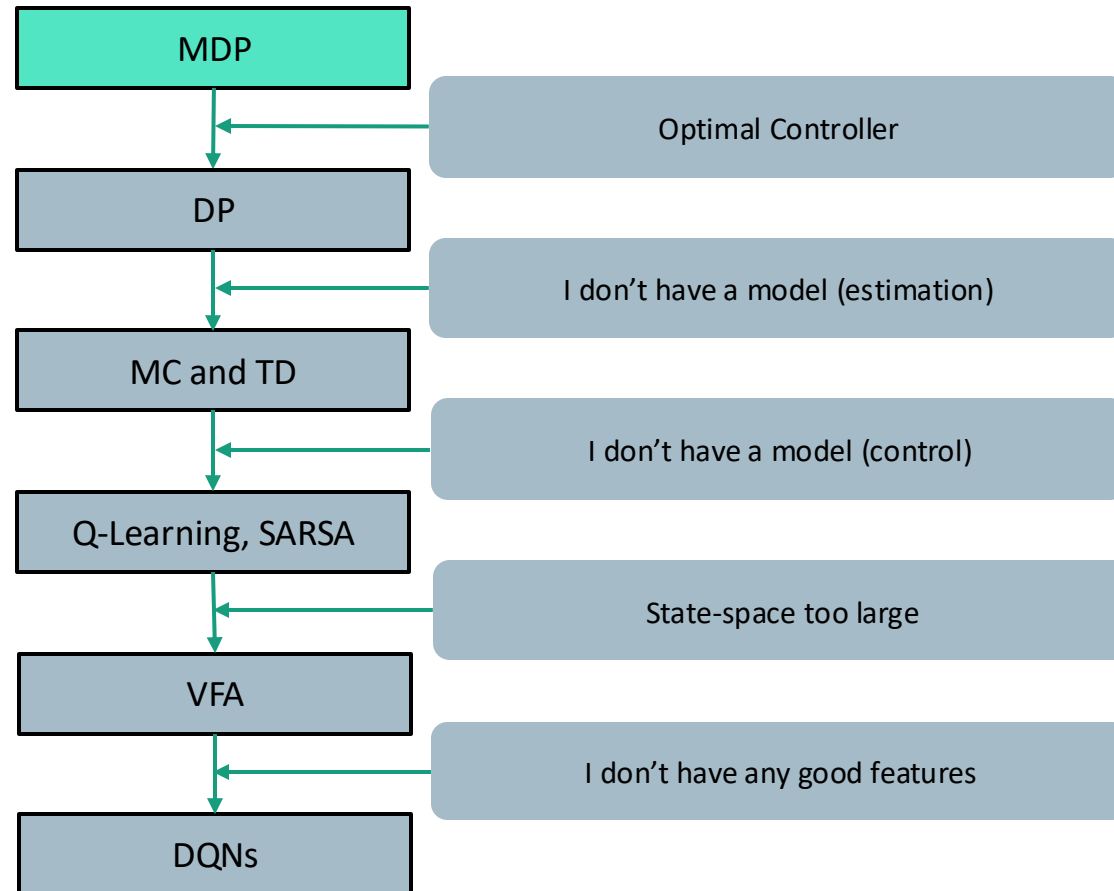


*Bertsekas, 2019:*

State of the art:

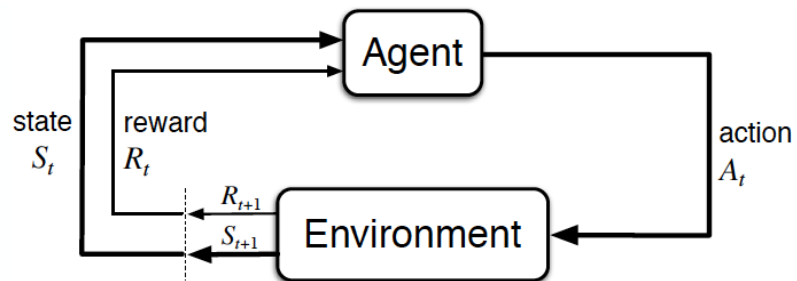
- **Broadly applicable methodology:** Can address a very broad range of challenging problems. Deterministic-stochastic-dynamic, discrete-continuous, games, etc
- There are **no methods that are guaranteed to work** for all or even most problems
- There are **enough methods to try with a reasonable chance of success** for most types of optimization problems
- **Role of the theory:** Guide the art, delineate the sound ideas

# Markov Decision Processes Overview



# Markov Decision Processes

- Agent learns by interacting with an environment over many time-steps:
- Markov Decision Process (MDP) is a tool to formulate RL problems
  - Description of an MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ :



Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

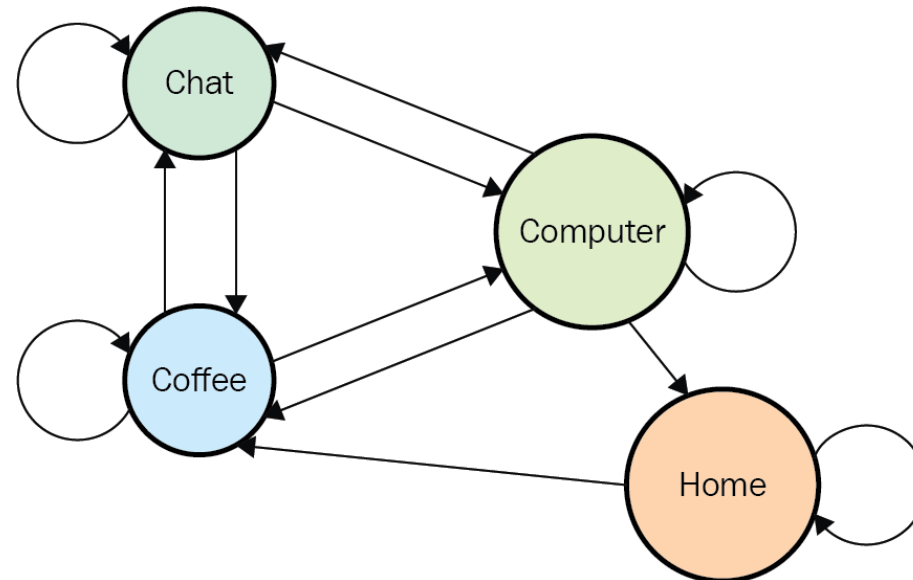
## Note:

If the interaction does stop at some point in time ( $T$ ) then we have an *episodic RL problem*.

- At each step  $t$ , the agent:
  - is at state  $S_t$ ,
  - performs action  $A_t$ ,
  - receives reward  $R_t$ .
- At each step  $t$ , the environment:
  - receives action  $A_t$  from the agent,
  - provides reward  $R_t$ ,
  - moves at state  $S_{t+1}$ ,
  - increments time  $t \leftarrow t + 1$ .

# Markov Decision Processes

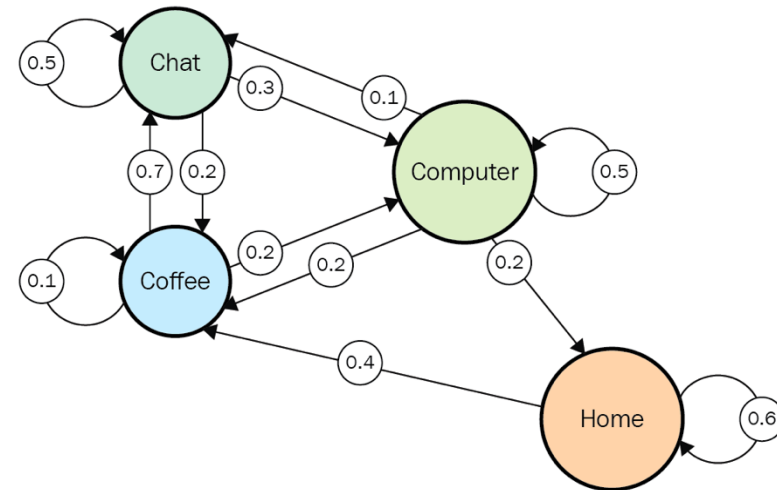
- Markov Process (MP)
  - Description of an MP ( $\mathcal{S}, \mathcal{P}$ ):



Lapan, M. (2018). *Deep Reinforcement Learning Hands-On*. Packt Publishing Ltd.

# Markov Decision Processes

- Markov Process (MP)
  - Description of an MP ( $\mathcal{S}, \mathcal{P}$ ):



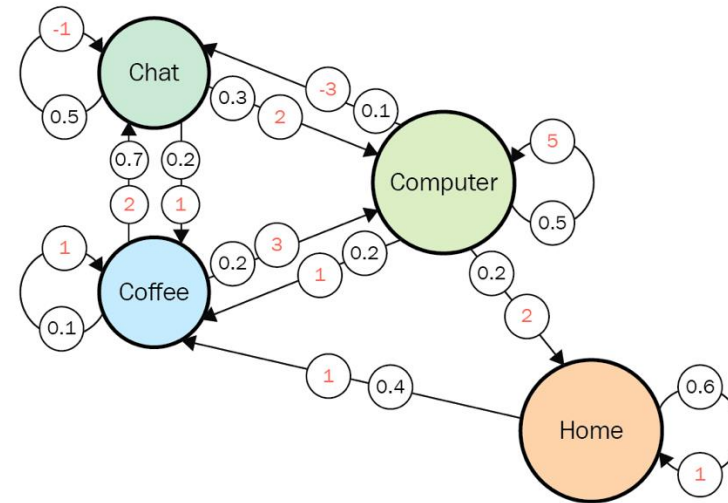
	Home	Coffee	Chat	Computer
Home	60%	40%	0%	0%
Coffee	0%	10%	70%	20%
Chat	0%	20%	50%	30%
Computer	20%	20%	10%	50%

Lapan, M. (2018). *Deep Reinforcement Learning Hands-On*. Packt Publishing Ltd.

# Markov Decision Processes

- Markov Reward Process (MRP)
  - Description of an MRP  $(\mathcal{S}, \mathcal{P}, \mathcal{R})$ :
  - $\mathcal{R}$  is a reward function:

$$\mathcal{R}_S = \mathbb{E}[R_{t+1} | S_t = s]$$

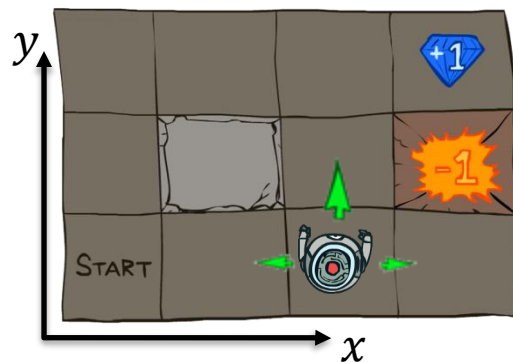


	Home	Coffee	Chat	Computer
Home	1	1		
Coffee		1	2	3
Chat		1	-1	2
Computer	2	1	-3	5

Lapan, M. (2018). *Deep Reinforcement Learning Hands-On*. Packt Publishing Ltd.

# Markov Decision Processes

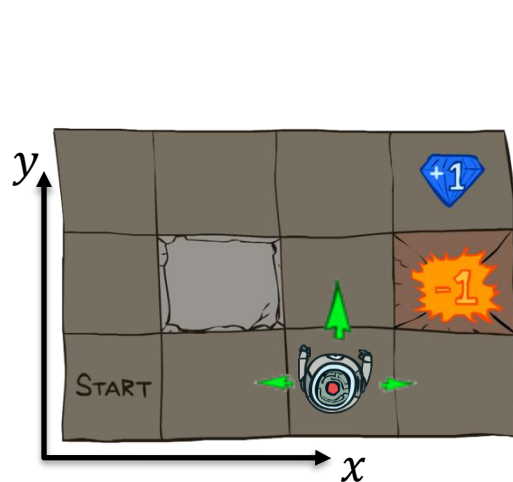
- Markov Decision Process (MDP)
  - Description of an MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ :



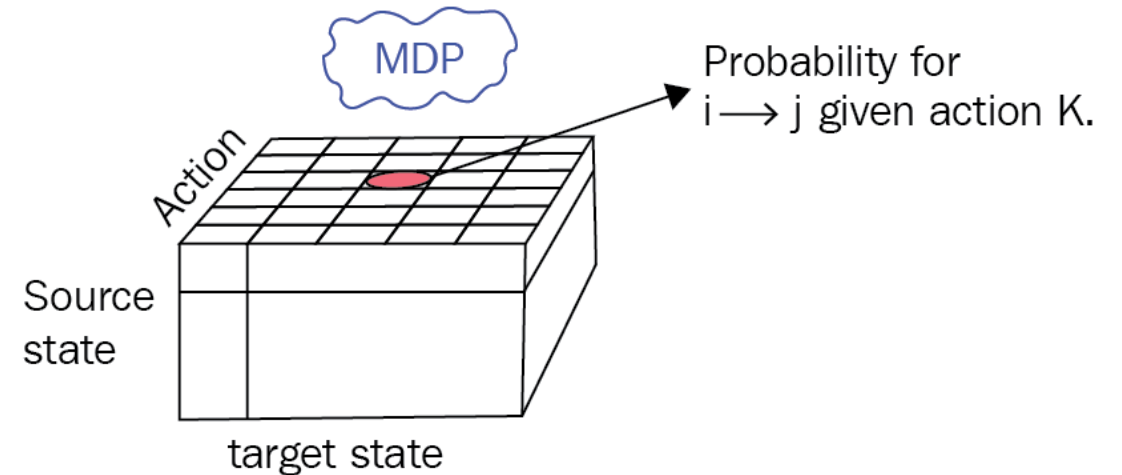
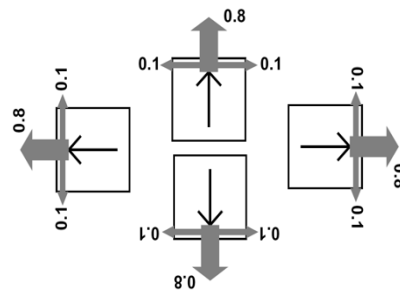
[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)

# Markov Decision Processes

- Markov Decision Process (MDP)
  - Description of an MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ :
  - State transition model:
    - A state transition probability matrix  $\mathcal{P}$  helps to model the true state transition function  $T(S_{t+1} | S_t, A_t)$  of a real-world environment.
    - For each action  $A^i \in \mathcal{A}$ , we have a state transition matrix  $\mathcal{P}^{A^i}$  at any time-step  $t$



[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)



Lapan, M. (2018). *Deep Reinforcement Learning Hands-On*. Packt Publishing Ltd.

# Markov Decision Processes

- Markov Decision Process (MDP)
  - Description of an MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ :
  - State transition model:
    - A state transition probability matrix  $\mathcal{P}$  helps to model the true state transition function  $T(S_{t+1} | S_t, A_t)$  of a real-world environment.
    - For each action  $A^i \in \mathcal{A}$ , we have a state transition matrix  $\mathcal{P}^{A^i}$  at any time-step  $t$  as follows:

**Notes:**

- Rows sum up to 1.0.
- $\mathcal{P}$  could change over time.

$$\begin{bmatrix} \mathcal{P}_{11} & \cdots & \mathcal{P}_{1n} \\ \vdots & \ddots & \vdots \\ \mathcal{P}_{n1} & \cdots & \mathcal{P}_{nn} \end{bmatrix}$$

# Markov Decision Processes

## about the state space $\mathcal{S}$

---

- History is the sequence of observations, actions, rewards:

$$H_t = O_0, A_0, R_0, O_1, A_1, R_1, O_2, \dots, O_{t-1}, A_{t-1}, R_{t-1}, O_t$$

- 3 different definitions of  $s_t$ :
  - (Full) Environmental state  $S_t^e$  (environment's private representation)**
    - Includes all the data that the environment uses to select next observation and reward
    - Private to the environment, not visible, maybe irrelevant information
  - Agent state  $S_t^a$  (agent's private representation; actually used)**
    - Private to the agent, history of observations, rewards, and actions
    - The agent constructs a state representation using a function of history  $S_t^a = f(H_t)$  to decide on the next action
  - Information state (useful information from the history)**
    - Basically,  $S_t^a$  with special constraints in  $f(H_t)$

# Markov Decision Processes

## about the state space $\mathcal{S}$

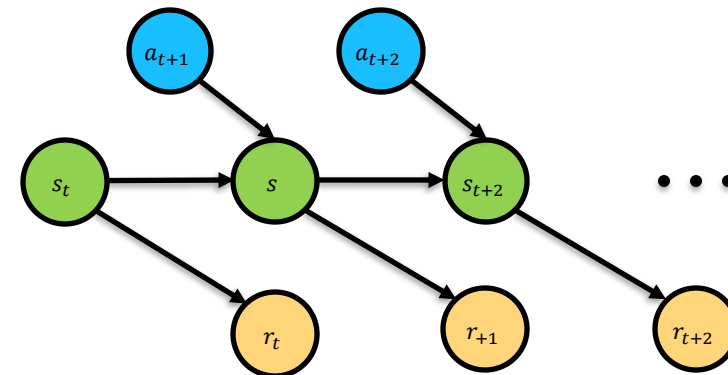
- Assumption of MDPs: Markov Property
  - A state  $S_t$  is Markov if and only if

$$\mathbb{P}[S_{t+1} | S_1, \dots, S_{t-1}, S_t] = \mathbb{P}[S_{t+1} | S_t]$$

- Past states  $S_1, \dots, S_{t-1}$  do not change the outcome for the next state  $S_{t+1}$ .
- The current state  $S_t$  captures all relevant information from the history.
- “The future is independent of the past given the present”**

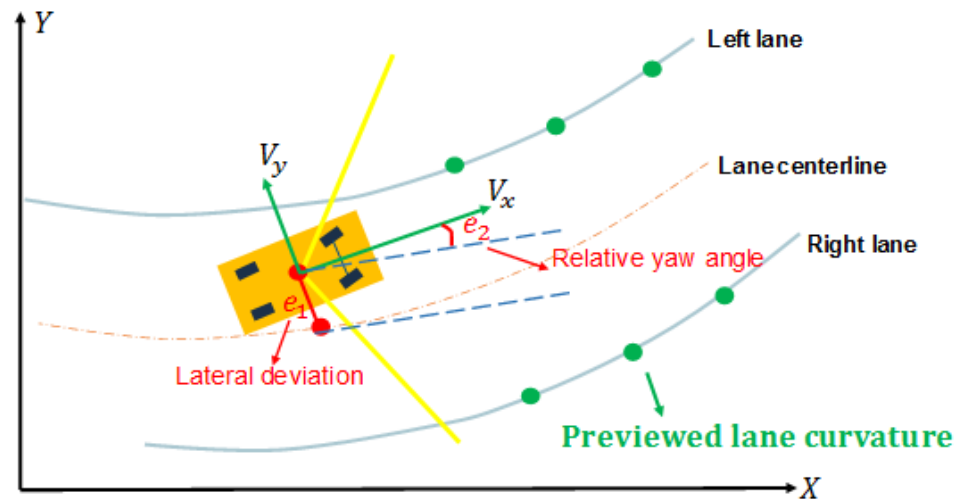
$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- State is the information used to determine what happens next
  - Direct (fully observable):  $O_t = S_t^e$
  - Indirect (partially observable):  $O_t = f(S_t^e)$



# Markov Decision Processes about the state space $\mathcal{S}$

- Assumption of MDPs: Markov Property
  - How can we ensure/construct such a Markov state?



Sensor Measurements:

- Speed, Angle

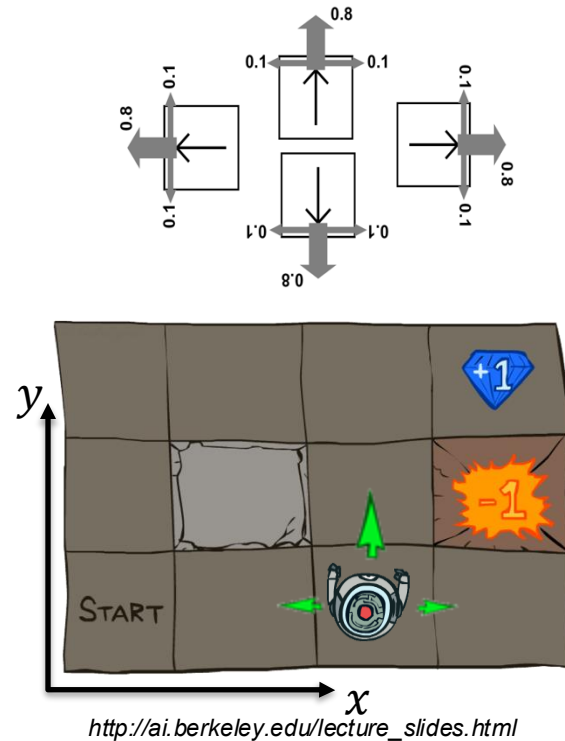
Requirements:

- Lateral acceleration
- Angular velocity

# Markov Decision Processes

## about the action space $\mathcal{A}$

- MDP example: Gridworld, episodic task

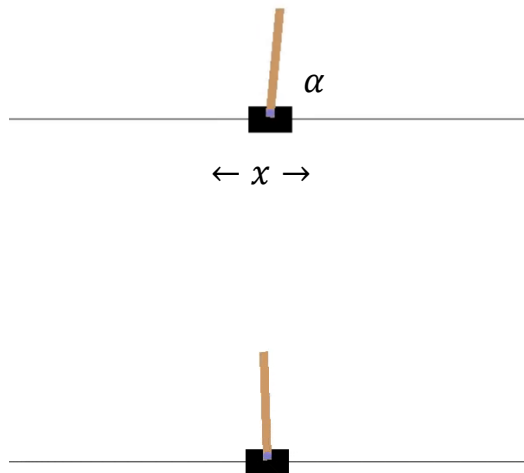


	Values
$\mathcal{S}$	$(x, y)$ with $x \in \{0, 1, 2, 3\}$ and $y \in \{0, 1, 2\}$
$\mathcal{A}$	LEFT, RIGHT, UP, DOWN,

# Markov Decision Processes ( $\mathcal{A}$ )

about the action space  $\mathcal{A}$

- MDP example: Cartpole, episodic or continuing task



	Values
$\mathcal{S}$	$(x, \theta, \dot{x}, \dot{\theta})$ with $x \in \mathbb{R}$ and $\alpha \in [0^\circ, 360^\circ]$
$\mathcal{A}$	LEFT, RIGHT

# Markov Decision Processes

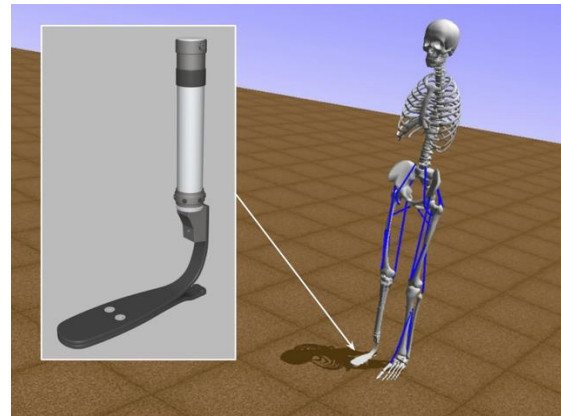
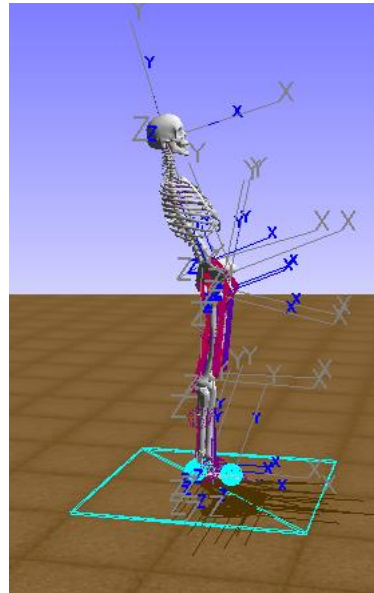
## MDP example: Tetris, *episodic task*

---



# Markov Decision Processes

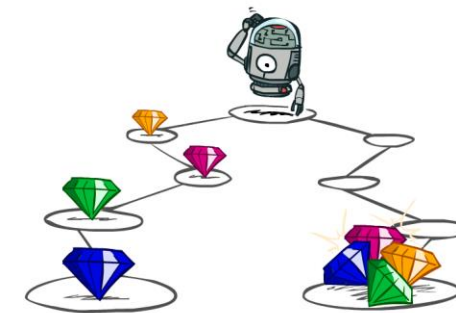
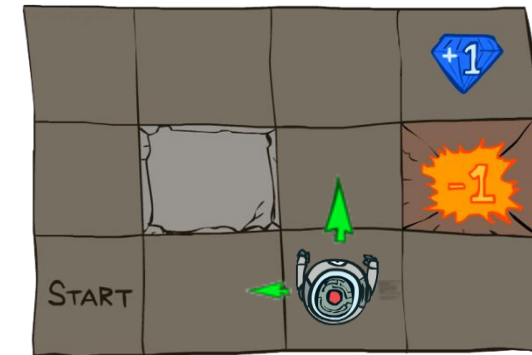
MDP example: Running with a prosthetic leg, *episodic task*



# of muscles	19
# degrees of freedom	14
reward	negative distance from requested velocity

# Markov Decision Processes

- Markov Decision Process (MDP) is a tool to formulate RL problems
  - Description of MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$
  - Recall: Actions have consequences!
  - Choosing an action  $A^i \in \mathcal{A}$  for  $A_t$  at timestep  $t$  yields different reward sequences
  - How do we know which sequence to prefer?
- Idea: Decay value of rewards over time.
  - $\gamma$  is a discount factor:  $\gamma \in [0,1]$



[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)

# Markov Decision Processes

- We want to “solve” the MDP, by maximizing future rewards.
  - We see the episodes in the form of

$$S_0 \xrightarrow{(A_0, R_0)} S_1 \xrightarrow{(A_1, R_1)} S_2 \xrightarrow{(A_2, R_2)} S_3 \dots S_{t-1} \xrightarrow{(A_{t-1}, R_{t-1})} S_t$$

- **Question:** what happens if our problem never stops (i.e.,  $T = \infty$ )?
  - Examples: data center cooling, recommender systems, etc.
- Total discounted ( $\gamma$ ) reward (**return**) (of one sample)

$$G = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots = \sum_{t=0}^{\infty} \gamma^t R_t$$

# Markov Decision Processes

---

- Markov Decision Process (MDP) is a tool to formulate RL problems
  - Description of MDP  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$
- **Why discount rewards with  $\gamma$ ?**
  - Mathematically convenient to discount rewards (true reason).
  - Avoids infinite returns in non-episodic problems
    - Datacenter cooling
    - Recommender system
  - Uncertainty about the future may not be fully represented (model uncertainty, our model is not perfect).
- Can I use  $\gamma = 1$ ?
  - Yes, if you have an episodic setting or you definitely know that there is a terminal absorbing state.
- Should I use  $\gamma = 1$ ?
  - **NO!**

# Markov Decision Processes

## about the policy $\pi$

---

- Expected long-term value of state  $s$ :

$$v(s) = \mathbb{E}(G) = \mathbb{E}(R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^t R_t)$$

- **Goal: maximize the expected return  $\mathbb{E}(G)$ .**
- We need a controller that helps us select the actions to maximize  $\mathbb{E}(G)$ !
- A policy  $\pi$  represents this controller:
  - $\pi$  determines the agent's behavior, i.e., its way of acting
  - $\pi$  is a mapping from state space  $\mathcal{S}$  to action space  $\mathcal{A}$

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

- Two types of policies:
  - Deterministic policy:  $a = \pi(s)$ .
  - Stochastic policy:  $\pi(a | s) = \mathbb{P}[A_t = a | S_t = s]$ .
- **New goal: find a policy that maximizes the expected return!**

# Markov Decision Processes

## Some remarks about terminology

---

$\mathbf{s}_t$  – state  
 $\mathbf{a}_t$  – action  
 $r(\mathbf{s}, \mathbf{a})$  – reward function



Richard Bellman

$$r(\mathbf{s}, \mathbf{a}) = -c(\mathbf{x}, \mathbf{u})$$

<http://rail.eecs.berkeley.edu/deeprcourse/static/slides/lec-2.pdf>

$\mathbf{x}_t$  – state  
 $\mathbf{u}_t$  – action  
 $c(\mathbf{x}, \mathbf{u})$  – cost function

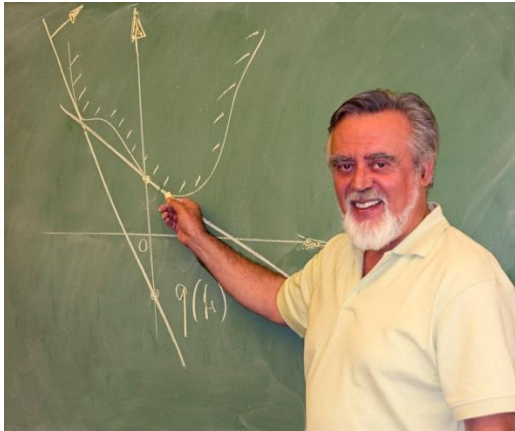


Lev Pontryagin

# Markov Decision Processes

## Some remarks about terminology

---



Bertsekas, 2019:

RL uses Max/Value, DP uses Min/Cost

- **Reward of a stage** = (Opposite of) Cost of a stage.
- **State value** = (Opposite of) State cost.
- **Value (or state-value) function** = (Opposite of) Cost function.

Controlled system terminology

- **Agent** = Decision maker or controller.
- **Action** = Decision or control.
- **Environment** = Dynamic system.

Methods terminology

- **Learning** = Solving a DP-related problem using simulation.
- **Self-learning (or self-play in the context of games)** = Solving a DP problem using simulation-based policy iteration.
- **Planning vs Learning distinction** = Solving a DP problem with model-based vs model-free simulation.