



# 强化学习原理及应用

## Reinforcement Learning (RL): Theories & Applications

Chao Yu (余超)

School of Computer Science and Engineering  
Sun Yat-Sen University

[based on David Silver and Sergey Levine's course]

- **Reinforcement Learning: An Introduction (Second Edition)**, Richard S. Sutton and Andrew G. Barto, MIT Press, Cambridge, MA, 2018.

2<sup>nd</sup> Edition Website:

<http://incompleteideas.net/sutton/book/the-book-2nd.html>

2<sup>nd</sup> Edition PDF version:

<http://incompleteideas.net/sutton/book/RLbook2018.pdf>

1<sup>st</sup> Edition html version:

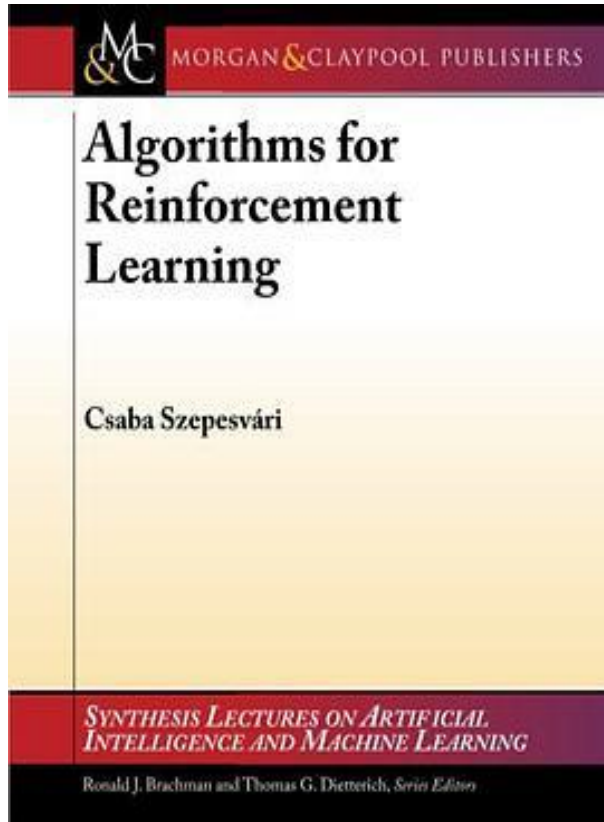
<http://incompleteideas.net/book/first/ebook/the-book.html>

## Reinforcement Learning

An Introduction  
second edition

Richard S. Sutton and Andrew G. Barto





**Algorithms for Reinforcement Learning,**  
Csaba Szepesvári,  
Morgan & Claypool Publishers, 2010.

<https://sites.ualberta.ca/~szepesva/rlbook.html>



**Reinforcement Learning State-of-the-Art,**  
Wiering M.A.,  
Springer, 2016.

# What is reinforcement learning?

*“The idea that we **learn by interacting with our environment** is probably the first to occur to us when we think about **the nature of learning**. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensori-motor connection to its environment. **Exercising this connection** produces a wealth of information about **cause and effect**, about the **consequences of actions**, and about **what to do in order to achieve goals**. Throughout our lives, such interactions are undoubtedly a major source of knowledge about our environment and ourselves. Whether we are learning to drive a car or to hold a conversation, we are all acutely aware of how our environment responds to what we do, and we seek to influence what happens through our behavior. **Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence.**”*

—Richard S. Sutton

*“Reinforcement learning problems involve learning what to do --- **how to map situations to actions** --- so as to **maximize a numerical reward signal**. In an essential way these are closed-loop problems because the learning system’s actions influence its later inputs. Moreover, the learner is **not told which actions to take**, as in many forms of machine learning, but instead must discover which actions yield the most reward by **trying them out**. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. **These three characteristics** --- **being closed-loop** in an essential way, **not having direct instructions** as to what actions to take, and where the consequences of actions, including reward signals, play out over **extended time periods** --- are the three most important distinguishing features of the reinforcement learning problem.”*

——Richard S. Sutton

RL, in a nutshell, is to “learn to *make good*  
*sequences of decisions through trial-and-errors*”

- Thus, there are four basic aspects in RL
  - Optimization (*good decisions*)
  - Delayed consequences (*sequential*)
  - Exploration (*trail-and-error*)
  - Generalization (*learn*)



- The goal is to find an optimal (or near-optimal) way to make decisions
- The evaluation of optimality can be explicitly measured or provided in terms of utility functions, e.g.,
  - *the shortest path between two cities given a network of roads*
  - *the fast speed that a robot is able to run*
  - *the maximum area for a multi-robot system to cover*
  - *the most money a gambling agent can win*
  - *the least time for a group of vehicles to pass a crossroad*
  - *the highest possibility to win a war*
  - ...

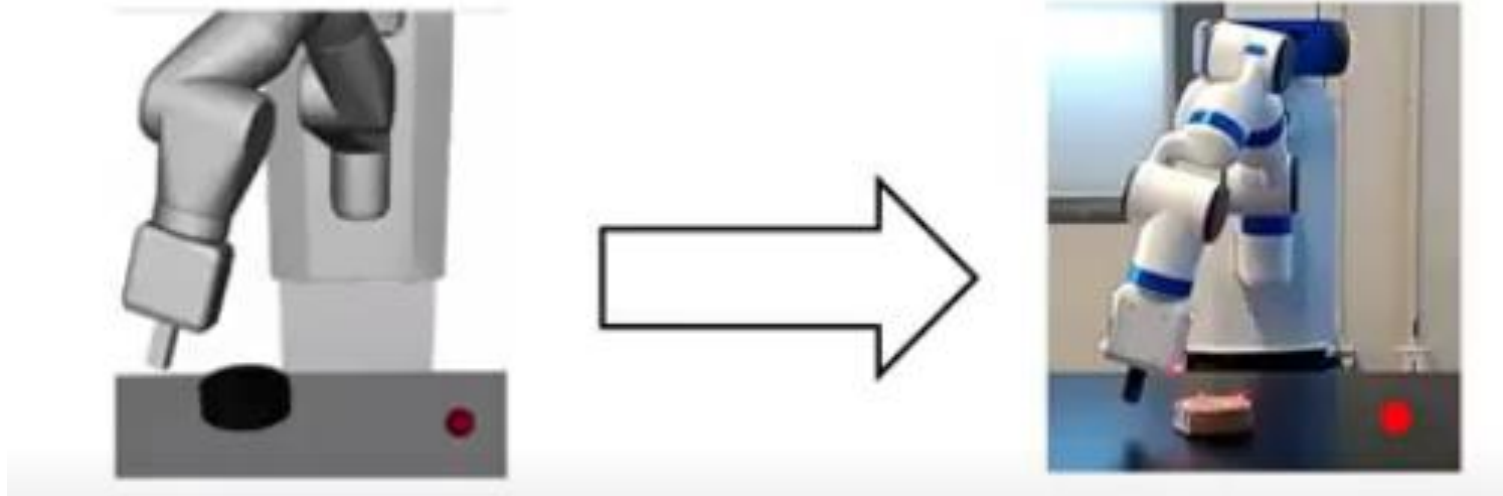
# Delayed consequences



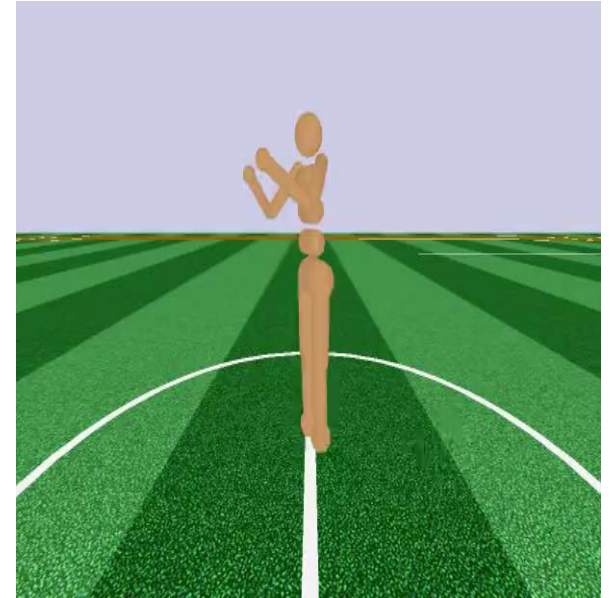
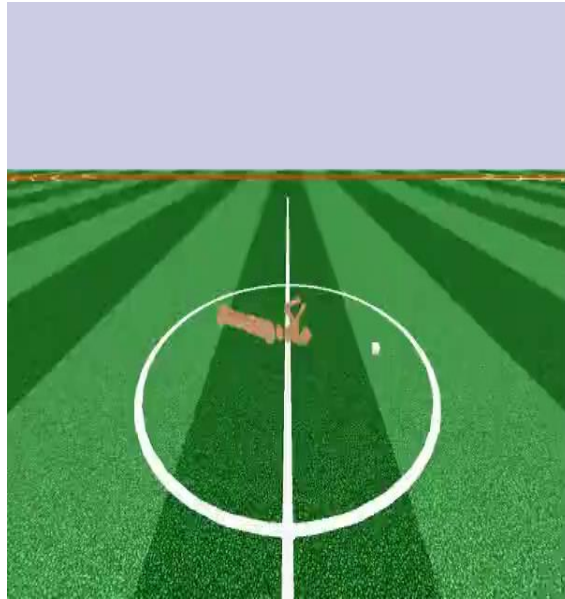
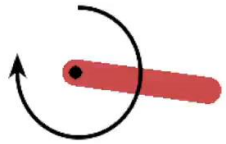
- Consequences of decisions can be much delayed
  - Actions will impact the input in the next step (data is not i.i.d. )
  - Should trade-off short-term and long-term outcome
- Cause the *credit assignment* problem
  - Different action sequences cause different outcomes
  - What caused later high or low rewards? Which action is responsible for the success or failure?

- RL is about learning from trial-and-error interactions
- Gain knowledge from the taken actions
  - *I know that restaurant A is good*
  - *I consider that the move of game playing is best*
- Trade-off between using the gained knowledge (exploitation) or try out new actions (exploration)
  - *Should I try other restaurant?*
  - *Should I try other moves?*
- Cause the *exploration-exploitation trade-off* problem
- Indicate sample efficiency and optimality of RL algorithms
  - *Use the least exploration to gain the optimal policies*
  - *Be able to jump out of local sub-optimal solution space*

- Why not just pre-program a policy?
  - No or limited knowledge of how the world functions
  - A policy that predesigned for a domain might not work in other related domains.



# RL cases



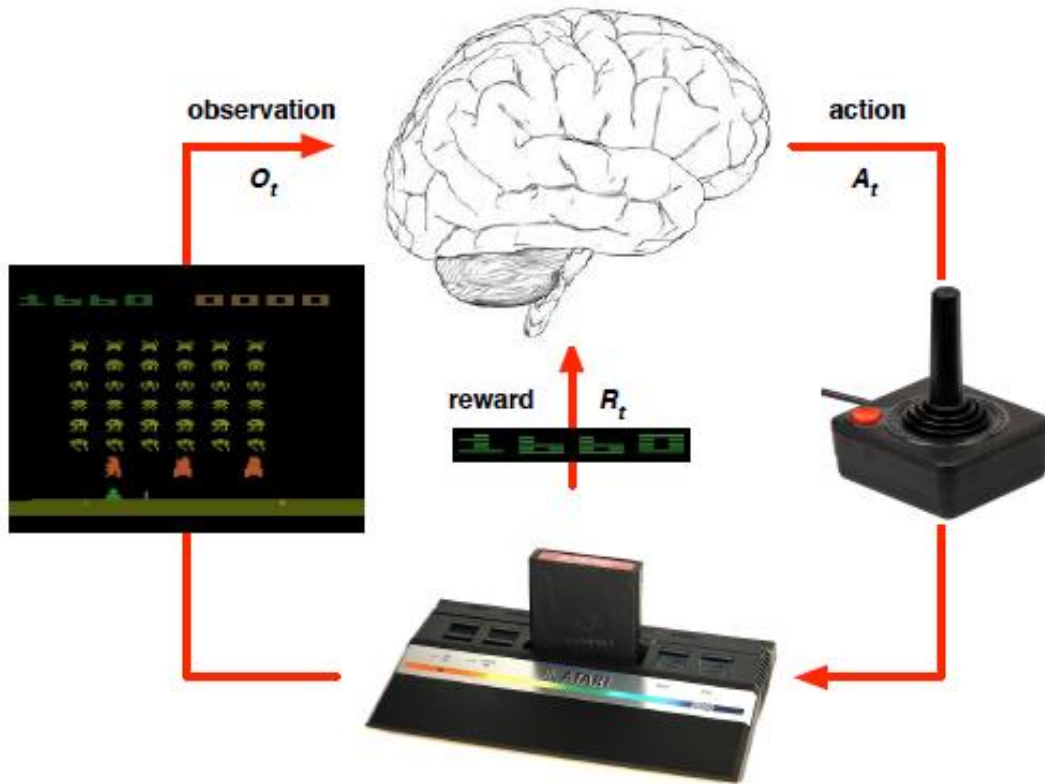
What makes RL different from other related paradigms?

- Two fundamental problems in **sequential decision making**:  
**learning** vs **planning**
- Reinforcement Learning:
  - *The environment is initially unknown*
  - *The agent interacts with the environment*
  - *The agent improves its policy*
- Planning
  - *A model of the environment is known*
  - *The agent performs computations with its model (without any external interaction)*
  - *The agent improves its policy*  
*a.k.a. deliberation, reasoning, introspection, pondering, thought, search*

# Characteristics of RL

What makes RL different from other related paradigms?

- Atari Example: RL



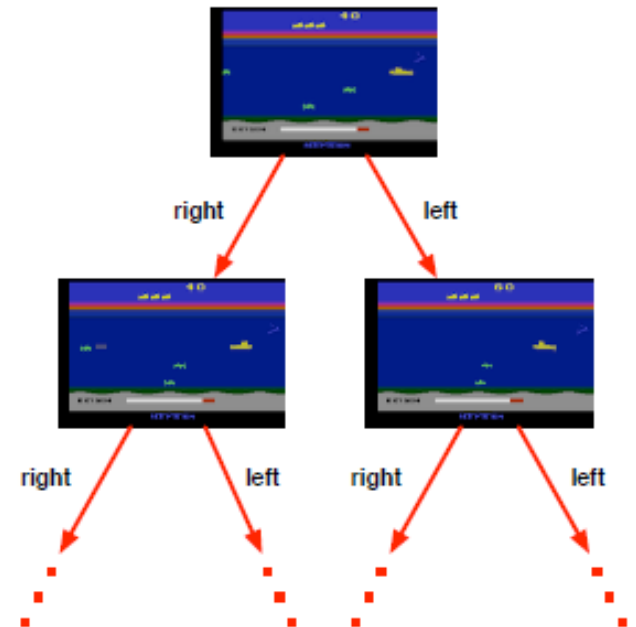
- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

# Characteristics of RL

What makes RL different from other related paradigms?

- Atari Example: Planning

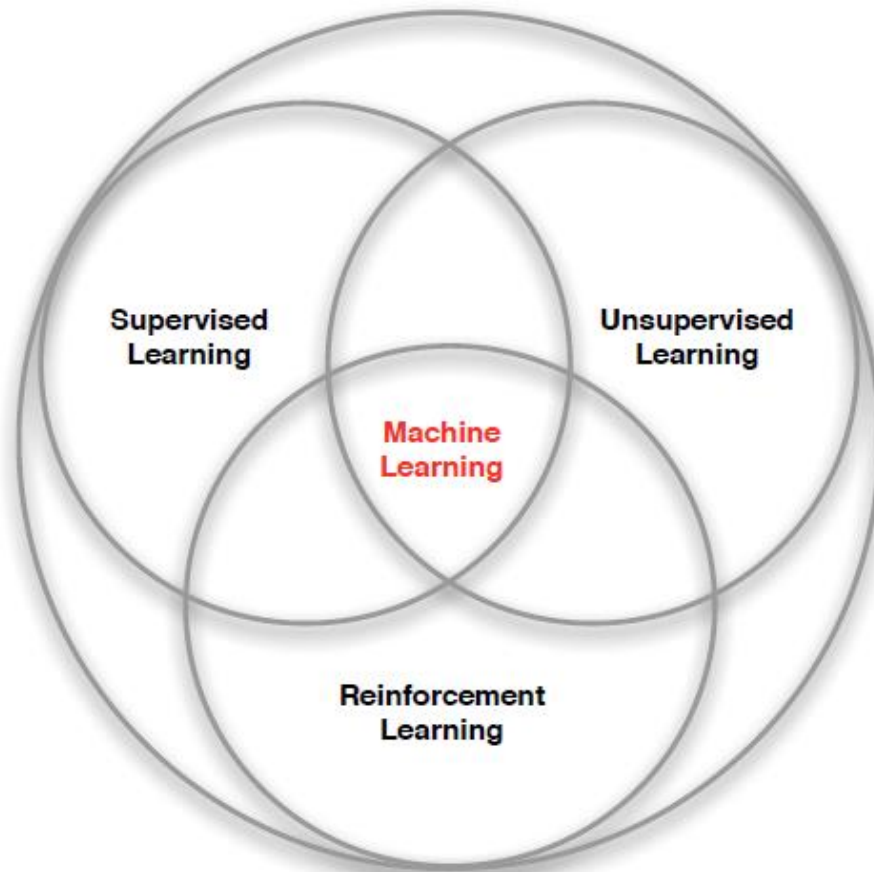
- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action  $a$  from state  $s$ :
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search





What makes RL different from other related paradigms?

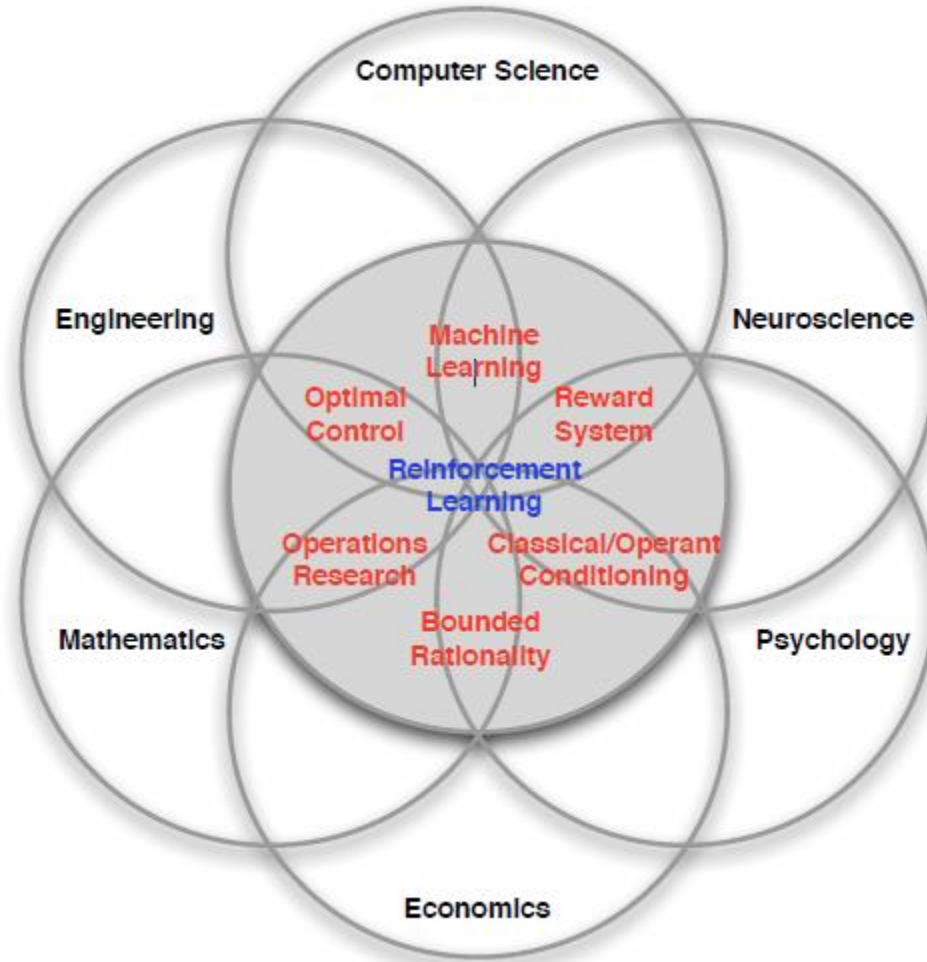
- Three fundamental problems in [machine learning](#)



What makes RL different from other related paradigms?

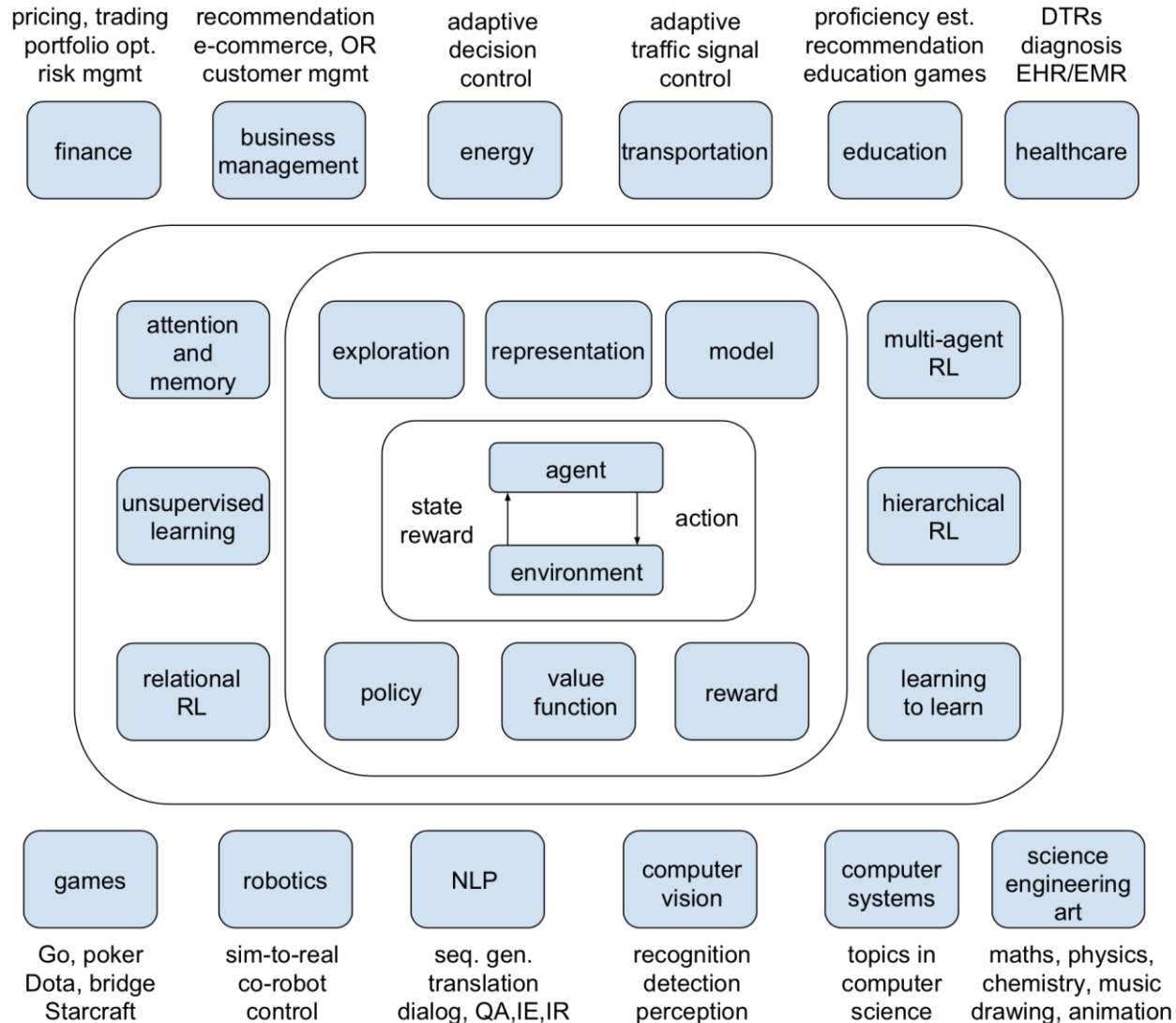
- Three fundamental problems in [machine learning](#)
- Supervised learning
  - *Classification or prediction from labeled (action,outcome) pairs*
  - *No interactions*
  - *No sequential decisions*
  - *No explorations*
- Unsupervised learning
  - *Discover inherent correlations among data*
  - *No interactions*
  - *No sequential decisions*
  - *No explorations*

# Interdisciplinary RL



[from David Silver's course]

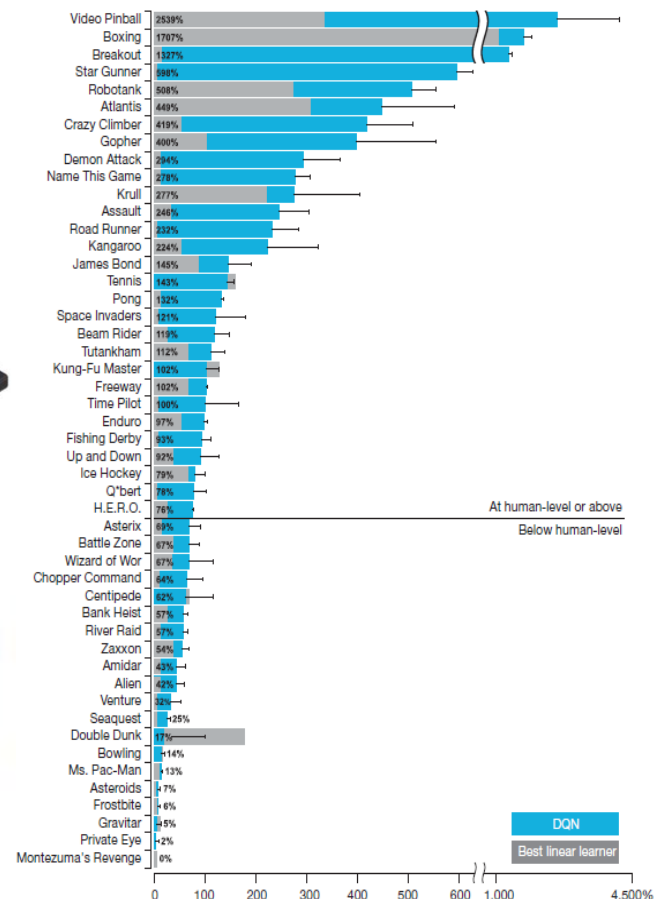
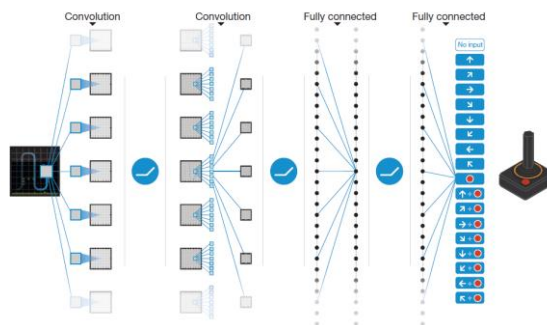
# Applications of RL



# Applications of RL—game playing

## Human-level Atari agent (2015)

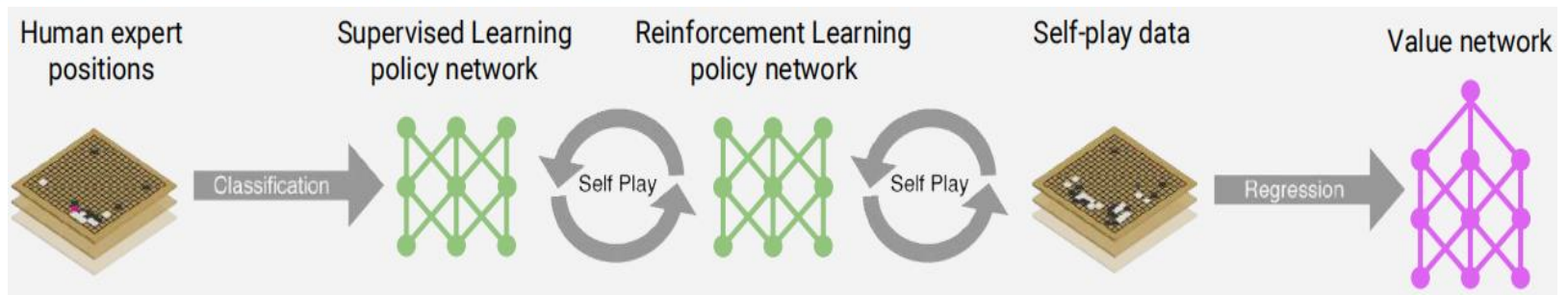
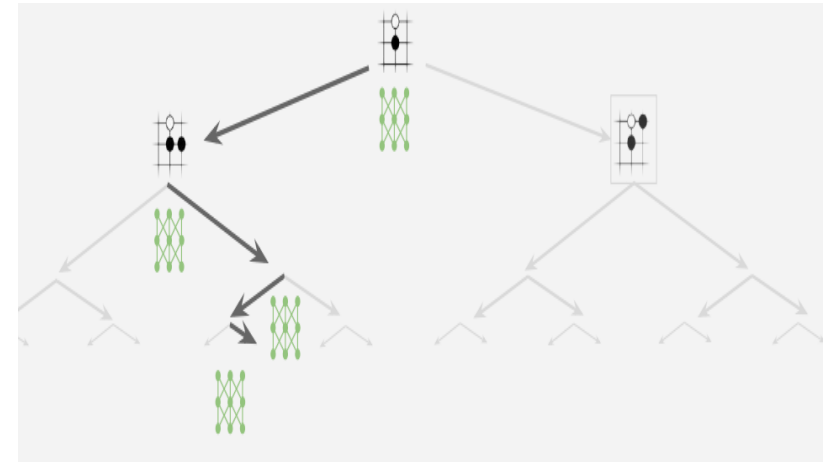
**Human-level control  
through deep reinforcement  
learning**



Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, 2015, 518(7540): 529-533.

# Applications of RL–game playing

## DeepMind's AlphaGo (2016)



Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529.7587 (2016): 484-489.



# Applications of RL—game playing



**Dota OpenAI Five**  
(OpenAI, 2018)

Defeat world champion



**StarCraft AlphaStar**  
(DeepMind, 2019, Nature)

Win professionals (top 10)

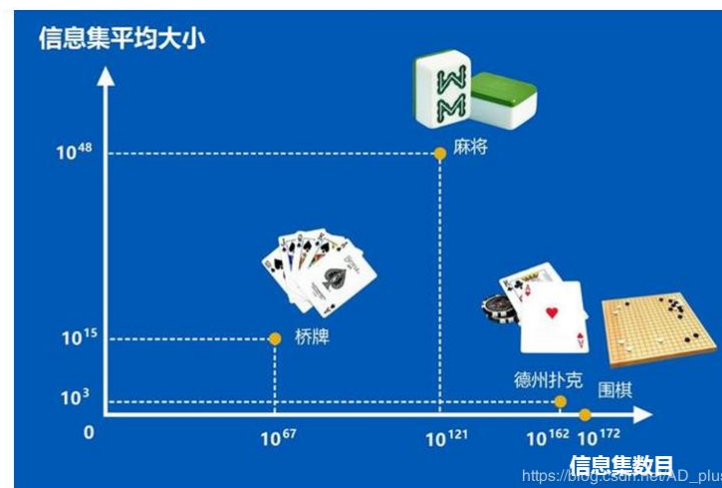


**Texas Hold'em Pluribus**  
(Facebook&CMU, 2019, Science)

Win \$1000/hour, i.e., \$5/hand

游戏	状态空间复杂度	游戏树复杂度
井字棋	$10^4$	$10^5$
国际跳棋	$10^{21}$	$10^{31}$
国际象棋	$10^{46}$	$10^{123}$
中国象棋	$10^{48}$	$10^{150}$
五子棋	$10^{105}$	$10^{70}$
围棋	$10^{172}$	$10^{360}$

[https://blog.csdn.net/AD\\_plus](https://blog.csdn.net/AD_plus)



宇宙原子数

$$8.64 \times 10^{-27} \times 4/3 \times \pi \times (4.3992 \times 10^{26})^3 \times 0.049 \times 0.75 / (1.6735 \times 10^{-27}) + 8.64 \times 10^{-27} \times 4/3 \times \pi \times (4.3992 \times 10^{26})^3 \times 0.049 \times 0.24 / (6.6465 \times 10^{-27}) \approx 7.31 \times 10^{79} \text{个}。$$

# Applications of RL –robot locomotion control





# Applications of RL –robot manipulation control



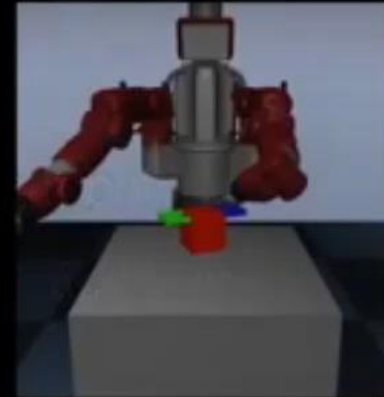
SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benc...



Block Lifting



Bimanual Peg-in-Hole



Bimanual Lifting



Block Stacking



Nut and Peg Assembly



Bin Picking



0:19 / 3:50

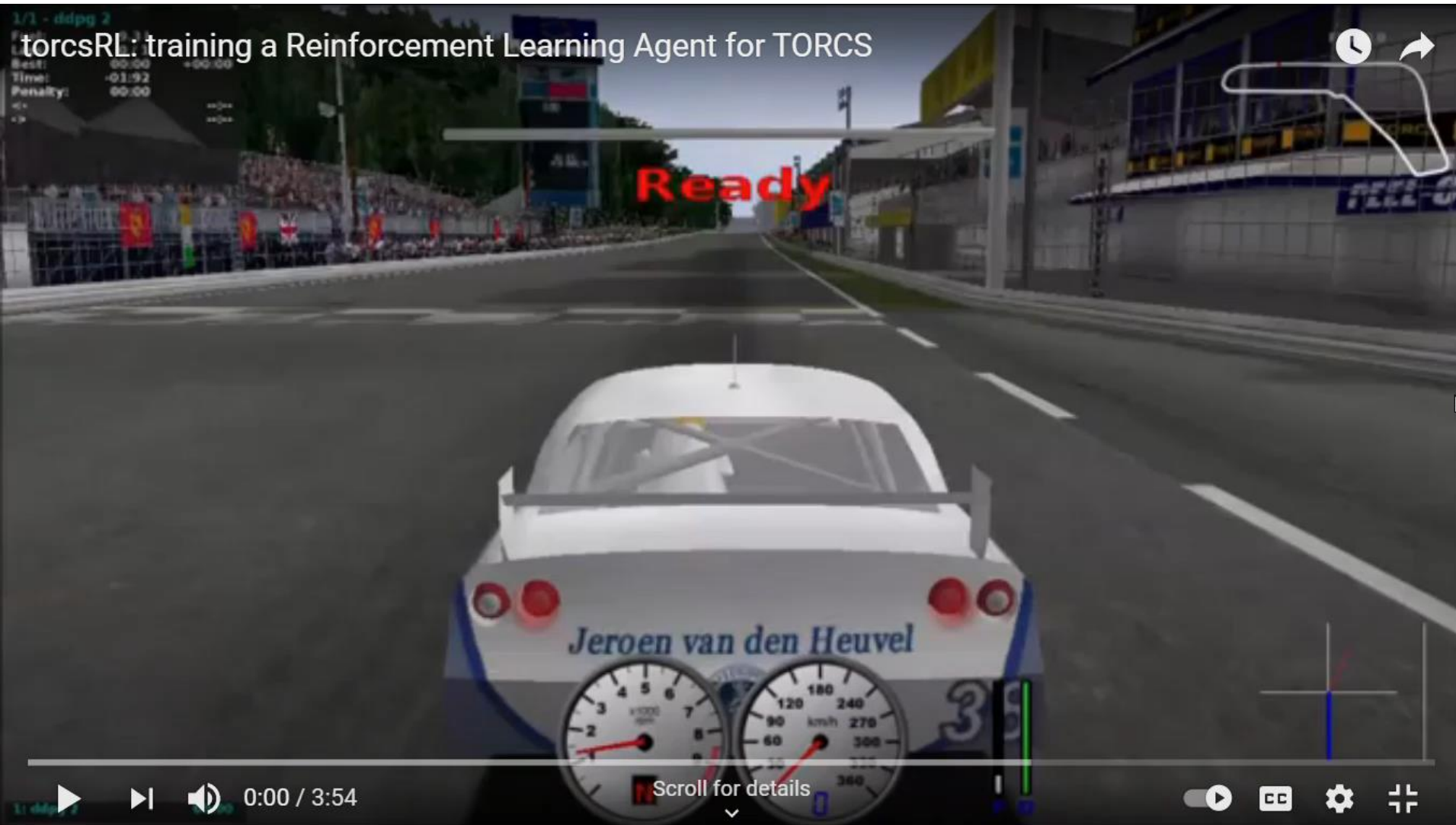
Scroll for details



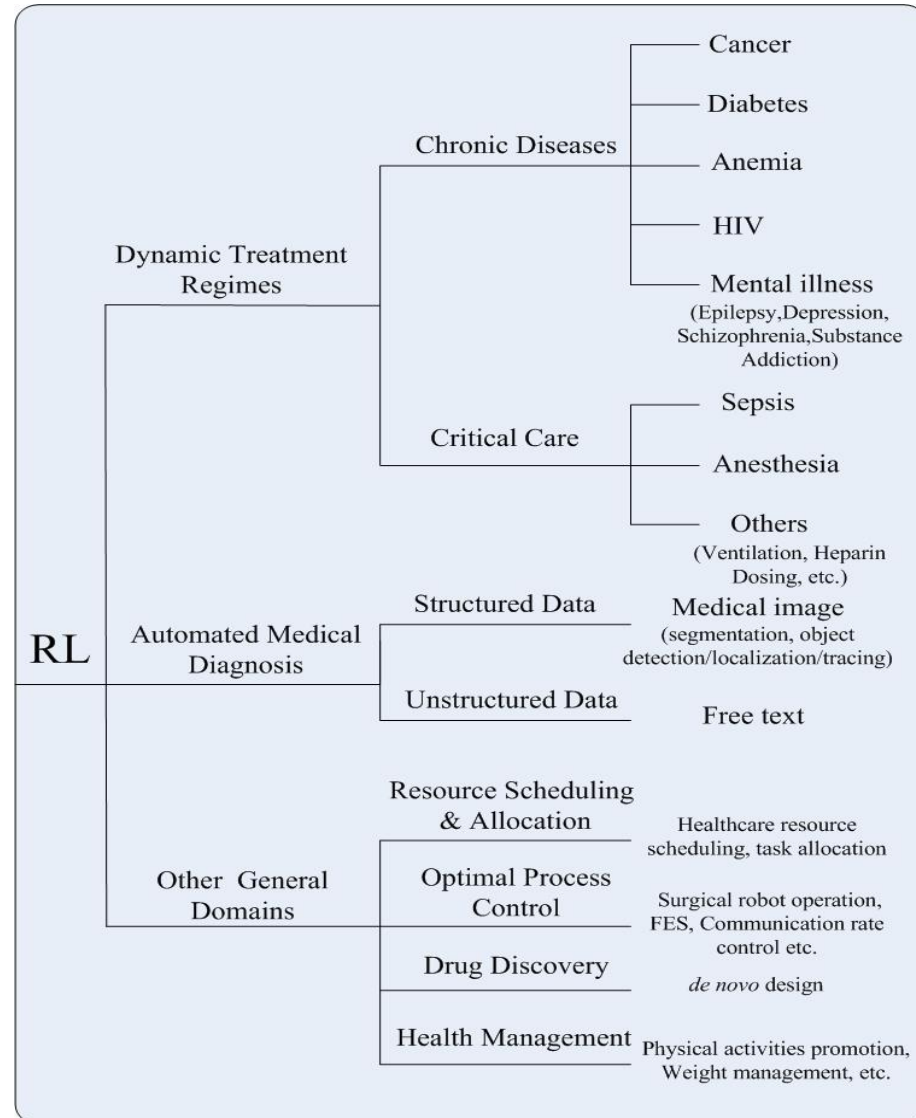
# Applications of RL –traffic control



# Applications of RL—autonomous driving



# Applications of RL-healthcare



## naturemedicine

[Explore Content](#) [Journal Information](#) [Publish With Us](#) [Subscribe](#)

[nature](#) > [nature medicine](#) > [comment](#) > [article](#)

[Comment](#) | [Published: 07 January 2019](#)

## Guidelines for reinforcement learning in healthcare

[Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez & Leo Anthony Celi](#) [✉](#)

[Nature Medicine](#) **25**, 16–18(2019) | [Cite this article](#)

**9928** Accesses | **33** Citations | **138** Altmetric | [Metrics](#)

**In this Comment, we provide guidelines for reinforcement learning for decisions about patient treatment that we hope will accelerate the rate at which observational cohorts can inform healthcare practice in a safe, risk-conscious manner.**

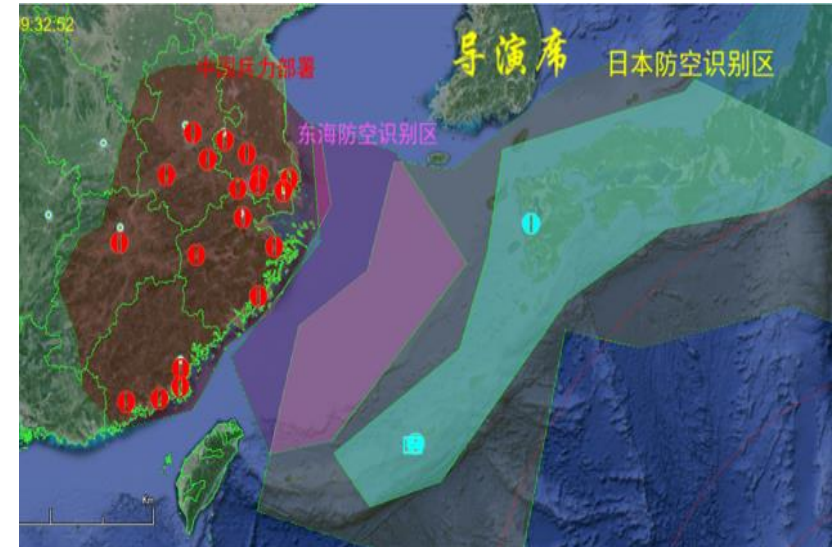
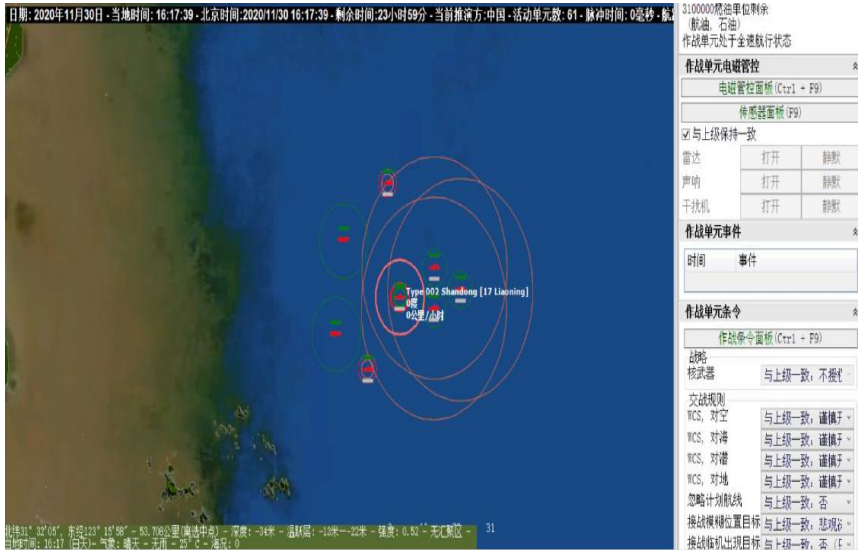
[Reinforcement learning in healthcare: A survey](#)

C Yu, J Liu, S Nemati, G Yin  
[ACM Computing Surveys \(CSUR\)](#) 55 (1), 1-36

132 2021



# Applications of RL—military



War game

# Applications of RL–military



Near-distance Air Combat

# Why reinforcement learning?

# Why RL is important?



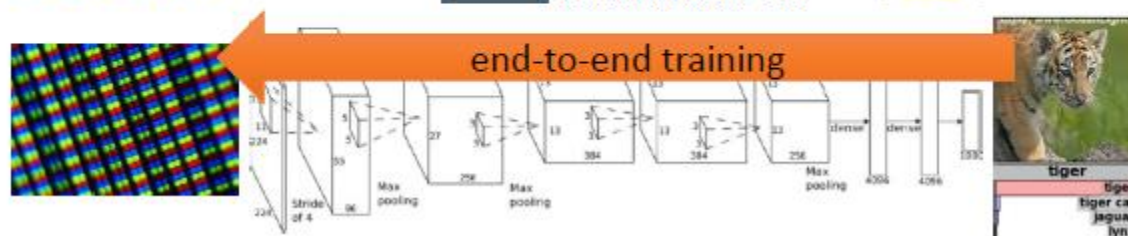
Fundamental challenge in artificial intelligence and machine learning is **learning to make good decisions under uncertainty**.



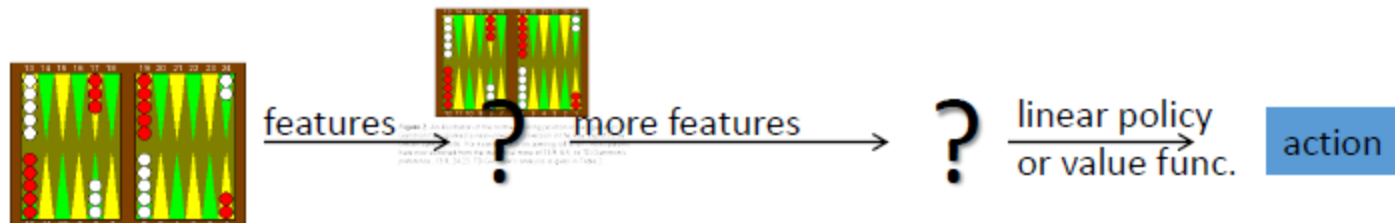
# Why RL is important?



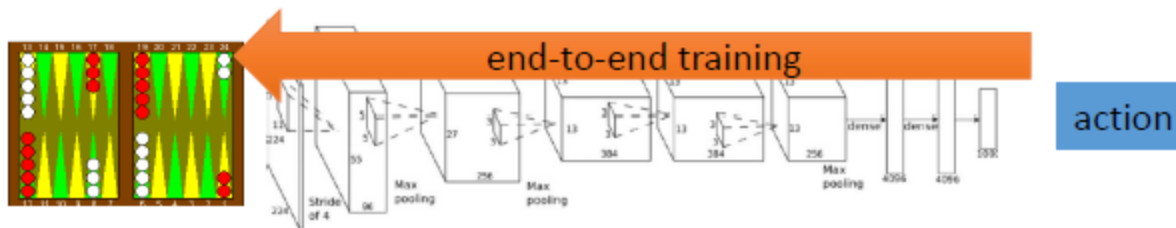
deep learning



standard reinforcement learning

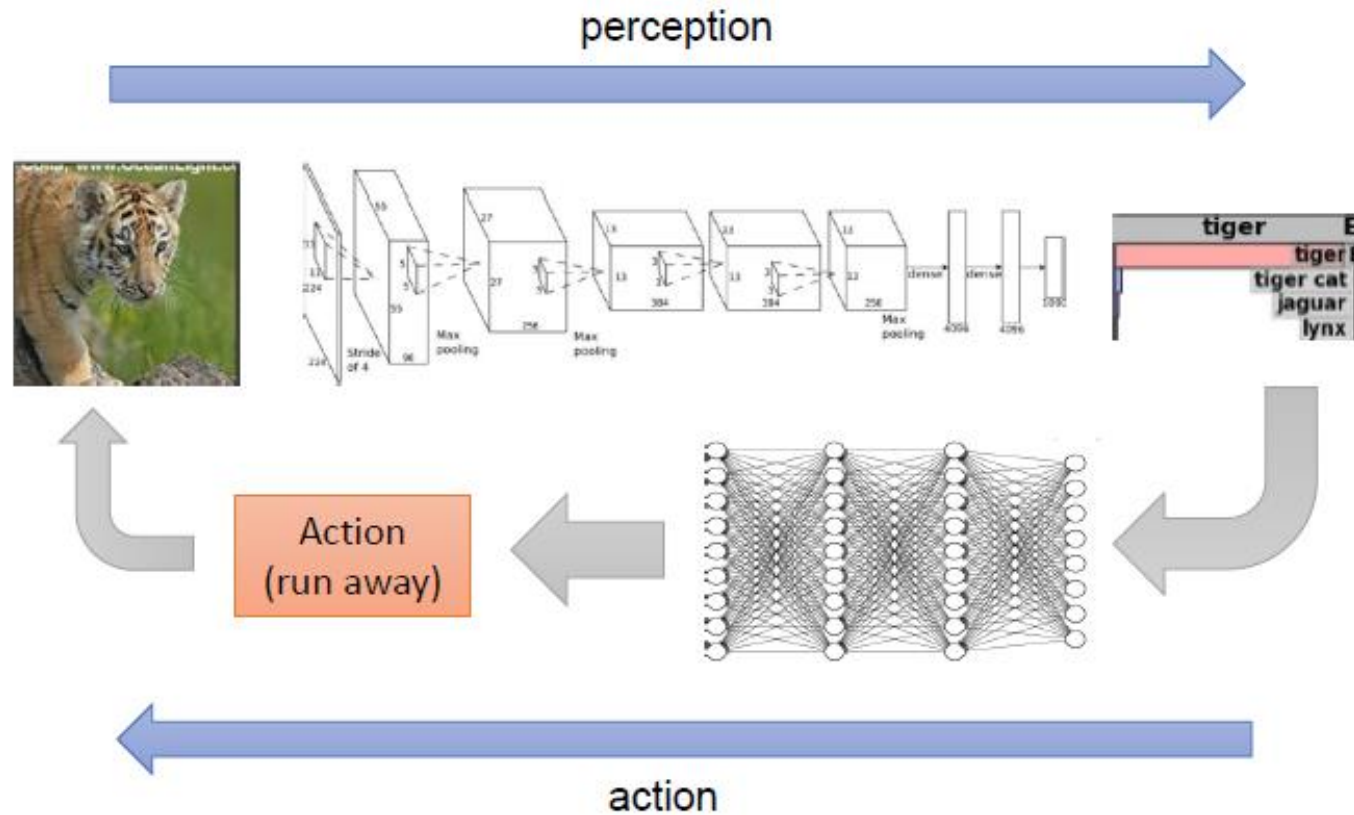


deep reinforcement learning



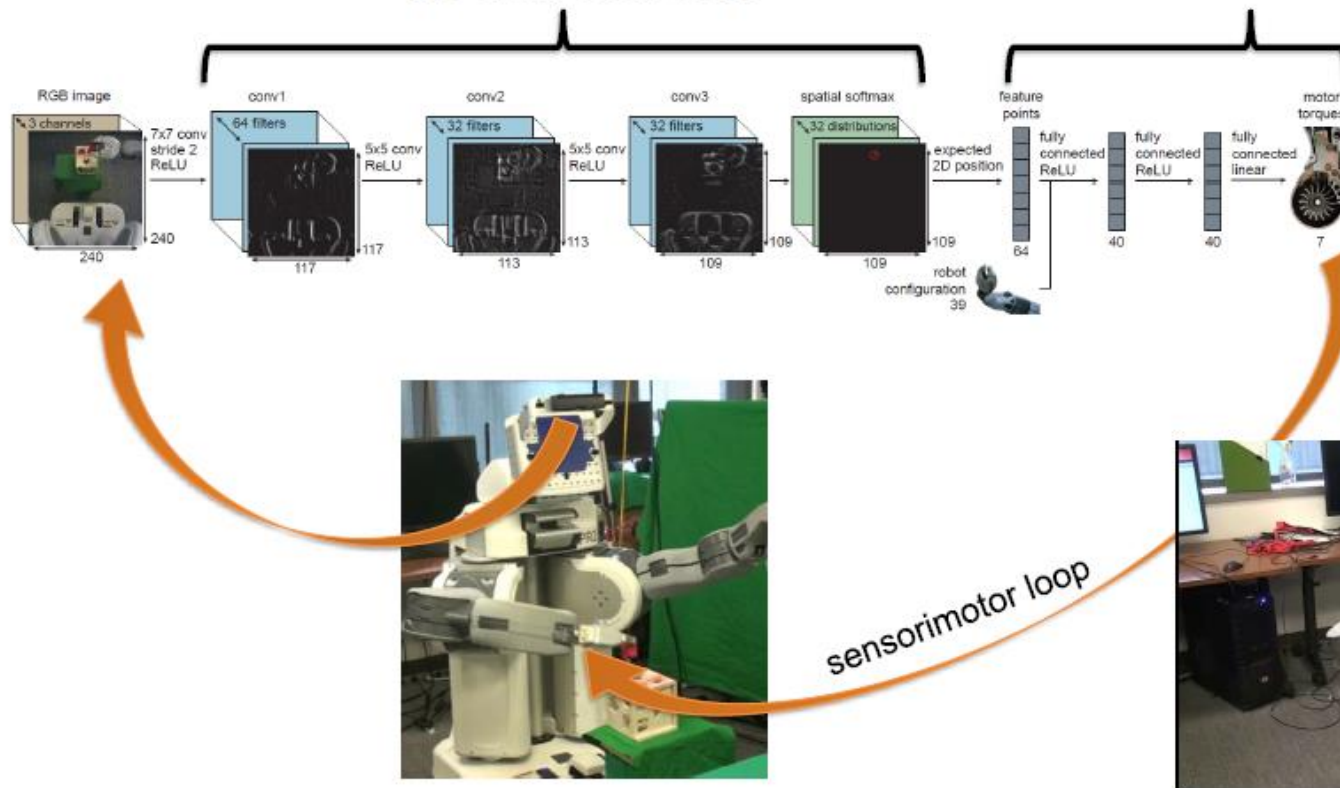
# Why RL is important?

RL closes the loop of decision making from perception to control



# Why RL is important?

tiny, highly specialized “visual cortex”      tiny, highly specialized “motor cortex”



The reinforcement learning problem **is** the AI problem!

[from Sergey Levine's course]

# Why now?

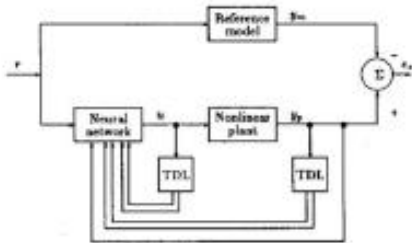


Fig. 21. Direct adaptive control of nonlinear plants using neural networks.

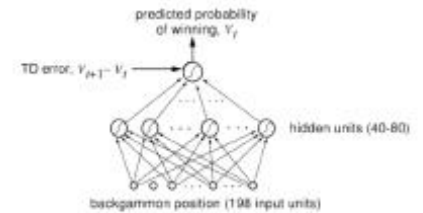
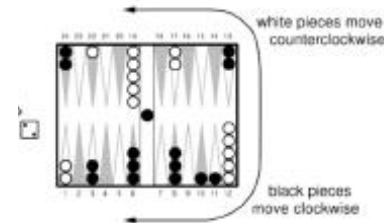
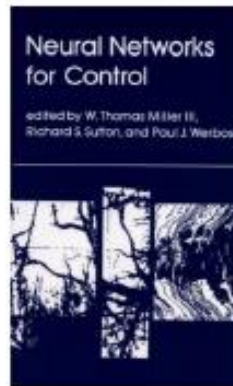


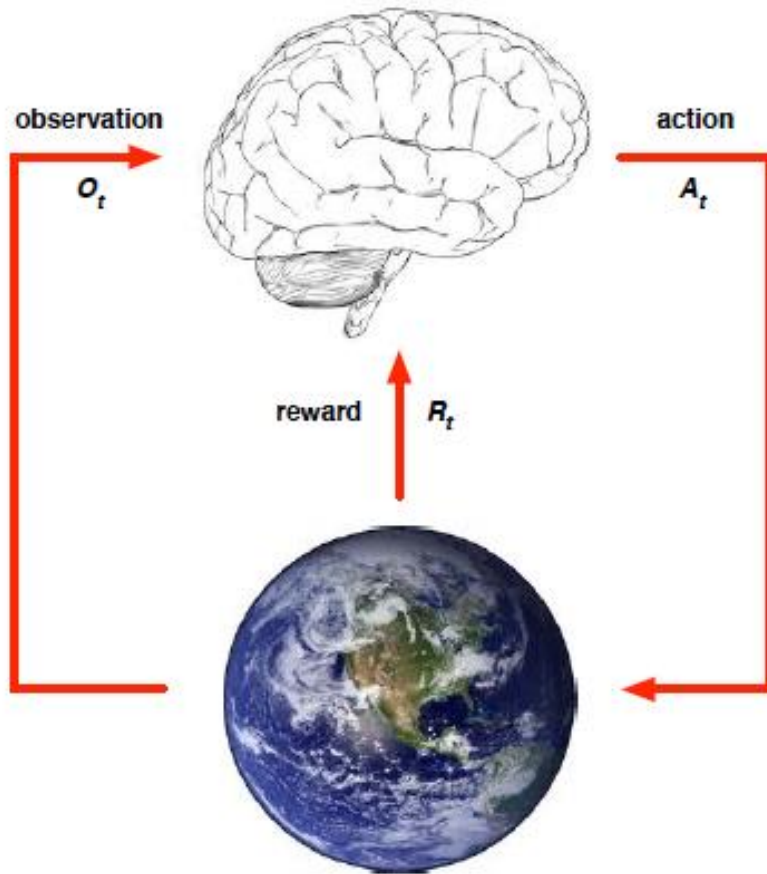
Table III.2: Summary of TD-Gammon Results

Program	Hidden Units	Training Games	Opponent	Results
TD-Gam 0.0	40	300,000	other programs	lost for test
TD-Gam 1.0	40	300,000	Roberta, Magik...	+ 5.3 ps / 15 games
TD-Gam 2.0	40	400,000	various Grandmasters	+ 7 ps / 50 games
TD-Gam 2.1	40	1,500,000	Roberta	+ 1 ps / 40 games
TD-Gam 3.0	40	1,500,000	Random	+ 14 ps / 30 games

Tesauro, 1995

- Advances in deep learning
- Advances in reinforcement learning
- Advances in computational capability

# The RL Problem

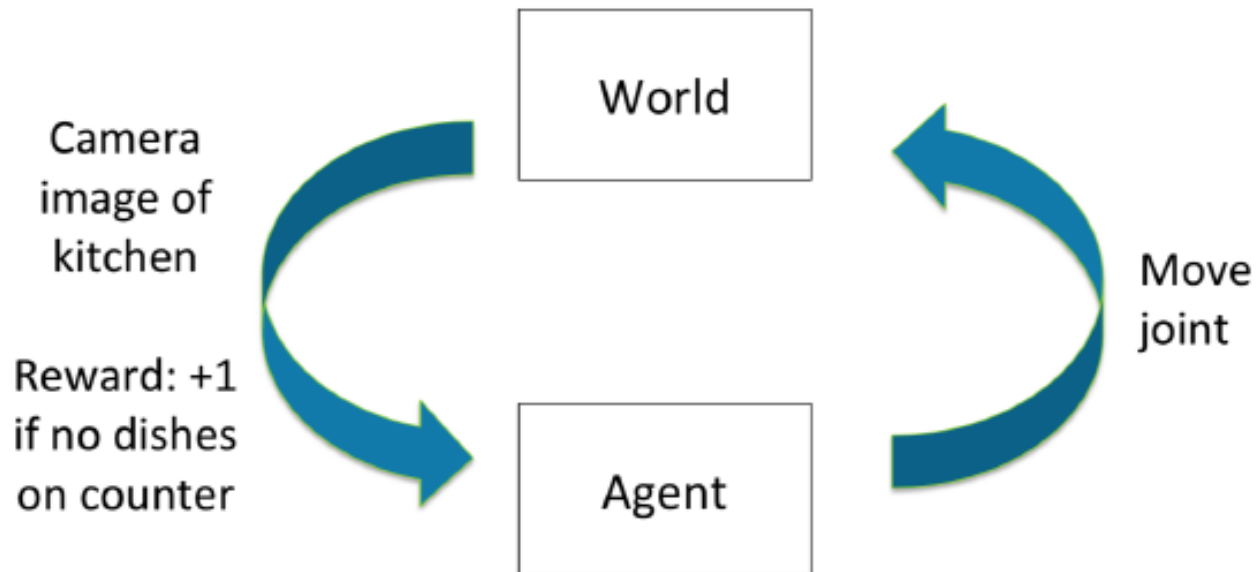


- At each step  $t$  the agent:
  - Executes action  $A_t$
  - Receives observation  $O_t$
  - Receives scalar reward  $R_t$
- The environment:
  - Receives action  $A_t$
  - Emits observation  $O_{t+1}$
  - Emits scalar reward  $R_{t+1}$

Goal: learn a policy (*i.e.*, a mapping from observations to actions) to maximise total future reward

# Example of RL Problems

## □ Robot Unloading Dishwasher

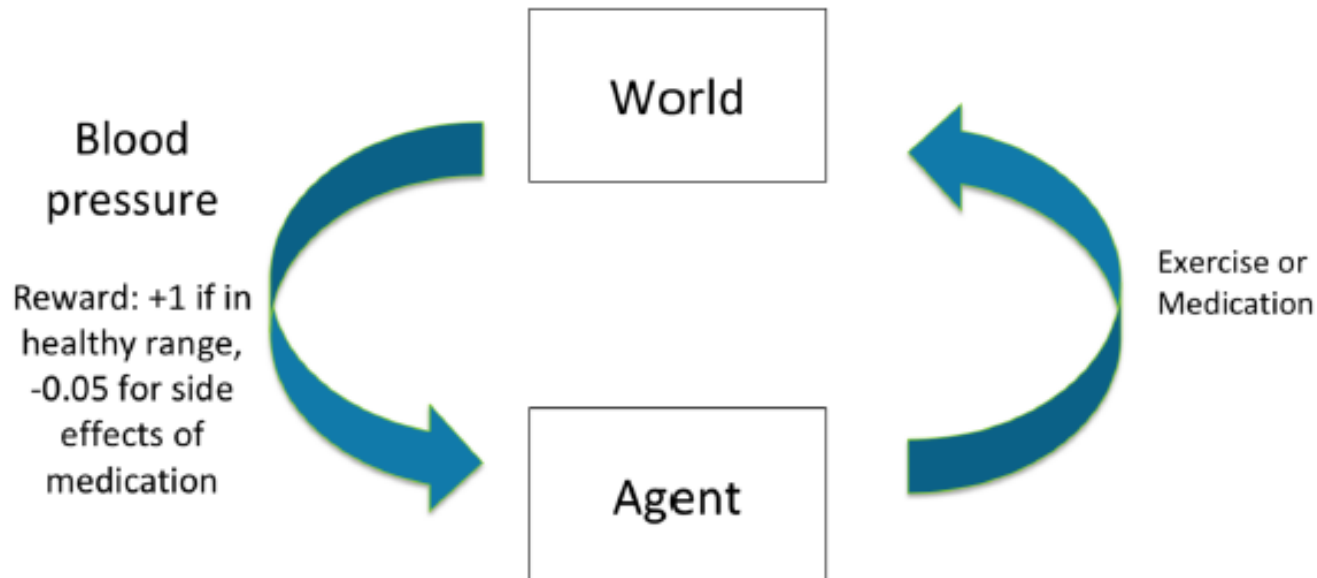


Goal: learn a policy (*i.e.*, a mapping from observations to actions) to maximise total future reward



# Example of RL Problems

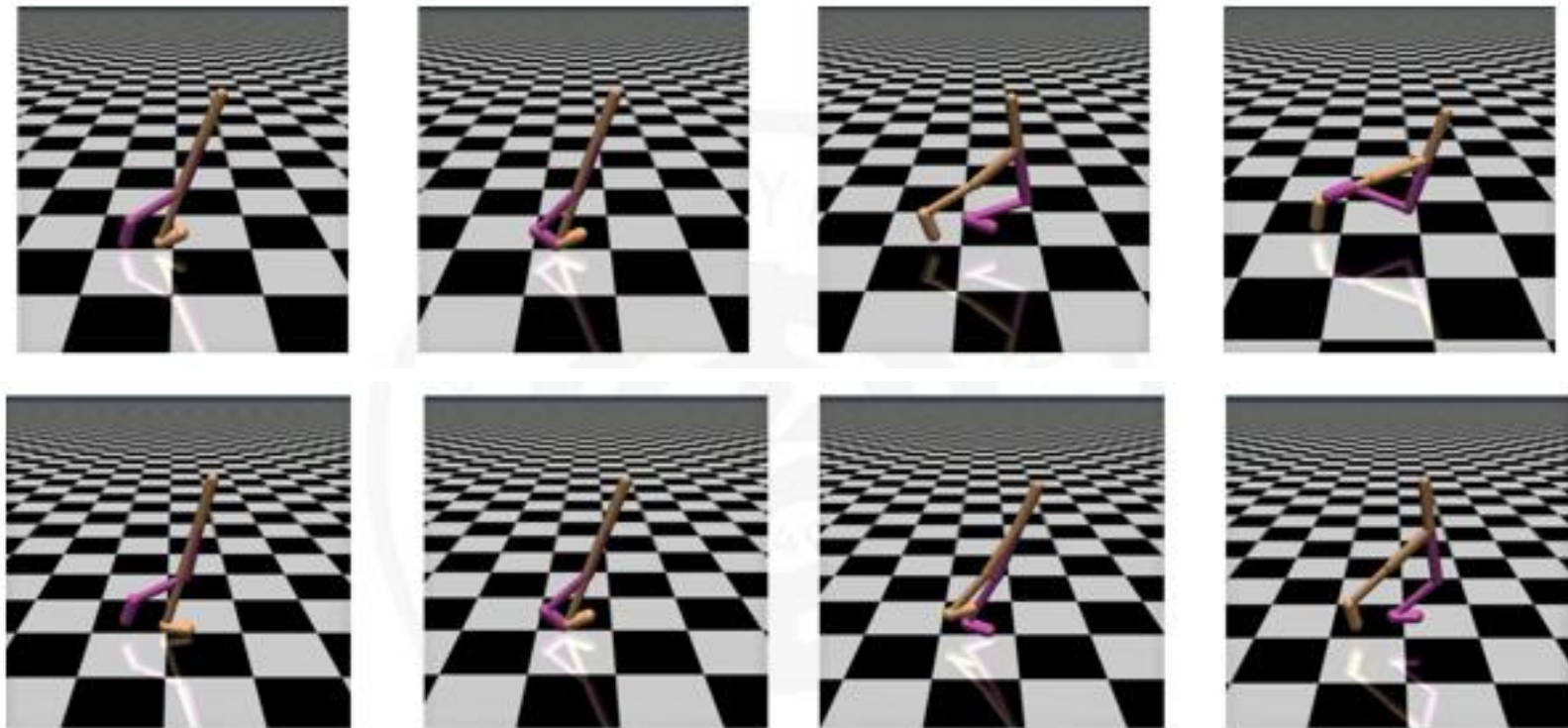
## □ Blood Pressure Control



Goal: learn a policy (*i.e.*, a mapping from observations to actions) to maximise total future reward

# Example of RL Problems

## ▣ Robotic control



Goal: learn a policy (*i.e.*, a mapping from observations to actions) to maximise total future reward!



# Elements of RL Problems - Reward

- ❑ A **reward**  $R_t$  is a scalar feedback signal
- ❑ Indicates how well agent is doing at step  $t$
- ❑ The agent's job is to maximise cumulative reward
- ❑ The goal reward and the intermediate reward
  - ❑ defeat the world champion at Go
    - +1/-1 reward for winning/losing a game
  - ❑ Make a humanoid robot walk
    - +1 reward for forward motion
    - 1 reward for falling over
  - ❑ Manage an investment portfolio
    - +v reward for each \$ in bank
- ❑ Reward is the most fundamental component in RL
  - ❑ Where is reward from? How to design the best reward? How to address sparse reward problems?
  - ❑ Inverse RL, Hierarchical RL, Transfer RL, Knowledge-driven RL, etc.

# Elements of RL Problems - State

- ❑ The **history** is the sequence of observations, actions, rewards
  - ❑ *i.e. all observable variables up to time  $t$*
  - ❑ *i.e. the sensorimotor stream of a robot or embodied agent*
- ❑ **State** is the information used to determine what happens next
- ❑ The environment state is its private representation
  - ❑ *whatever data to pick the next observation/reward*
  - ❑ *not usually visible to the agent*
  - ❑ *May contain irrelevant information*
- ❑ The agent state is the agent's internal representation
  - ❑ *whatever information the agent uses to pick the next action*
  - ❑ *it is the information used by RL algorithms*
- ❑ An Markov state contains all useful information from the history, i.e., future is independent of past given present

A state  $S_t$  is **Markov** if and only if

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

# Elements of RL Problems - State

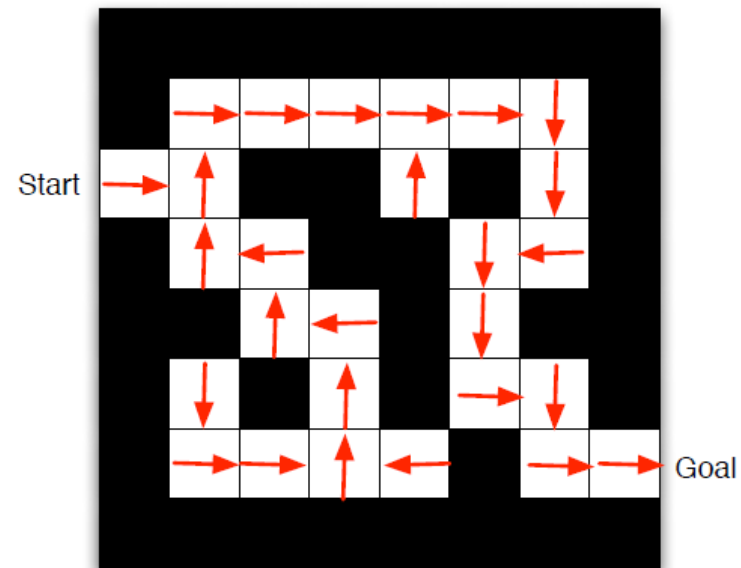
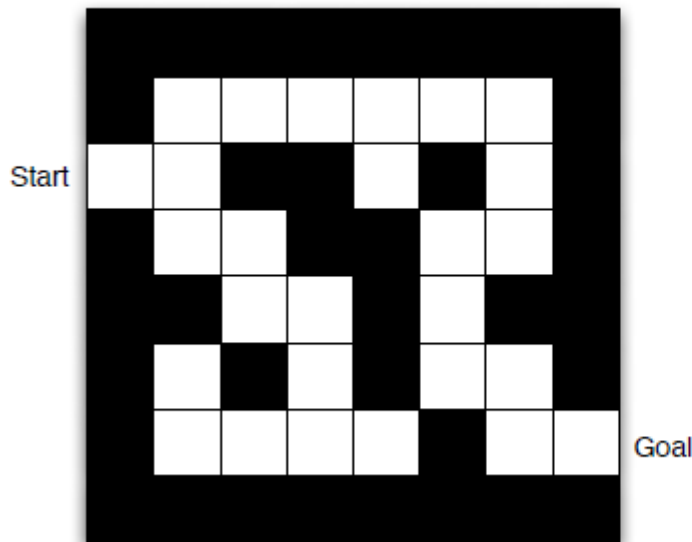
- ❑ **Full observability**: an agent directly observes environment state. Formally, this is a Markov decision process (MDP).
- ❑ **Partial observability**: an agent indirectly observes the environment, e.g.,:
  - ❑ *A robot with camera vision isn't told its absolute location*
  - ❑ *A trading agent only observes current prices*
  - ❑ *A poker playing agent only observes public cards*
  - ❑ *Formally, this is a partially observable Markov decision process (POMDP)*

# Elements of RL Problems - Policy

□ **Policy**: an agent's behaviour function, i.e., a mapping from state to action

□ Deterministic policy:  $a = \pi(s)$

□ Stochastic policy:  $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

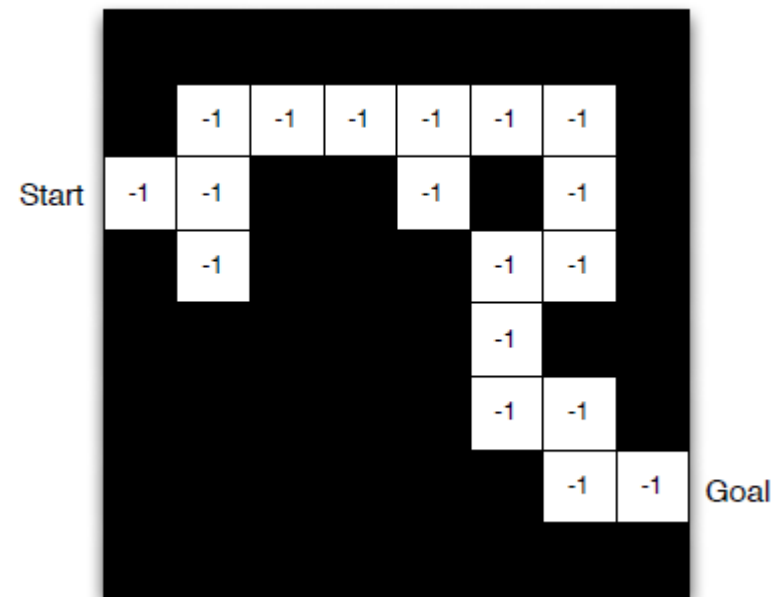


# Elements of RL Problems - Model

- ❑ **Model**: A model predicts what the environment will do next, i.e., agent's representation of the environment
- ❑  $P$  predicts the next state
- ❑  $R$  predicts the next (immediate) reward

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

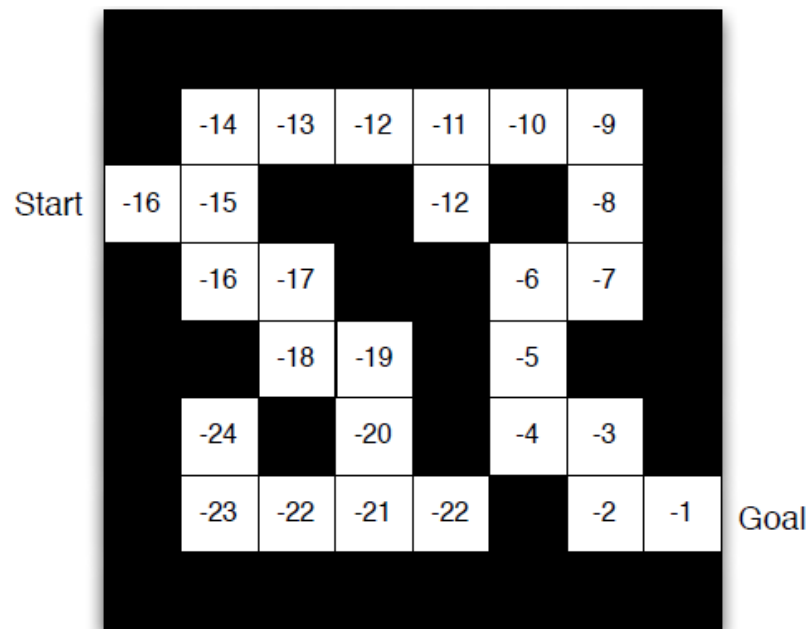


# Elements of RL Problems – Value Function



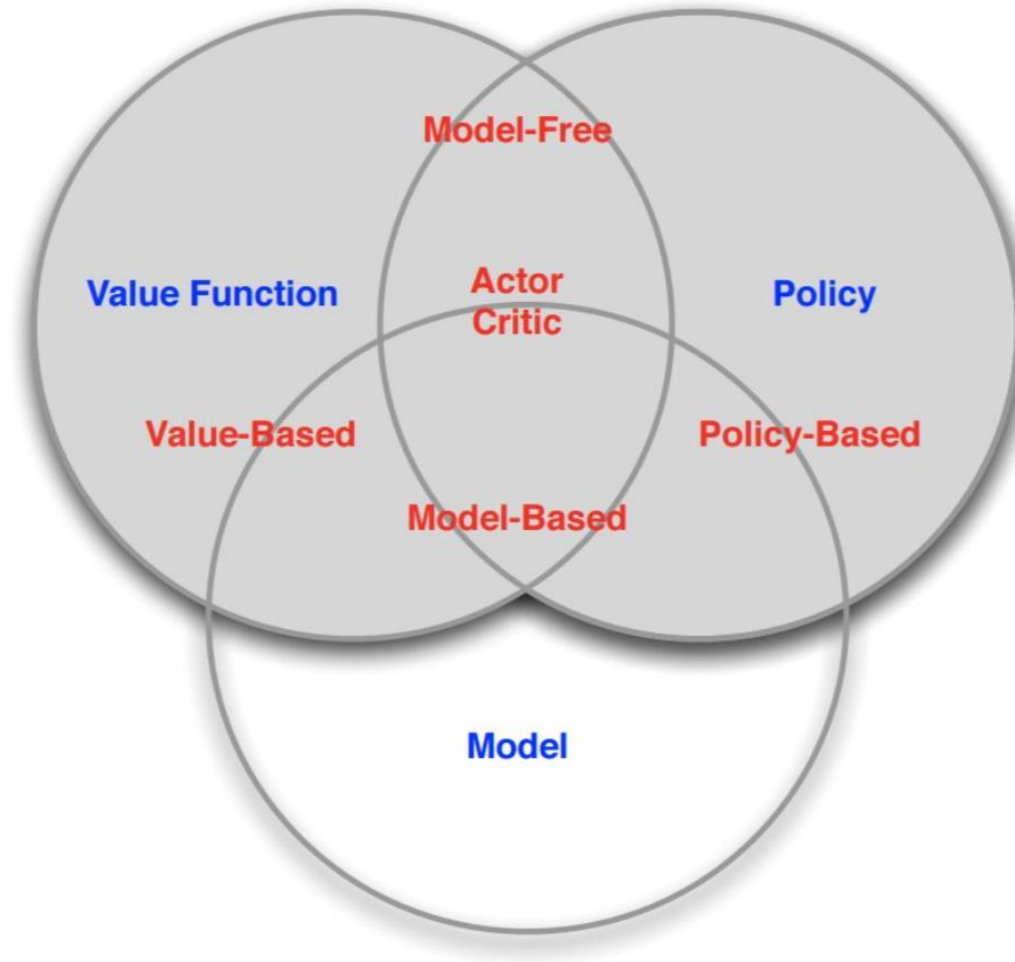
- Value functions: how good is each state and/or action
  - *Value function is a prediction of future reward*
  - *Used to evaluate the goodness/badness of states*
  - *And therefore used to select between actions*

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



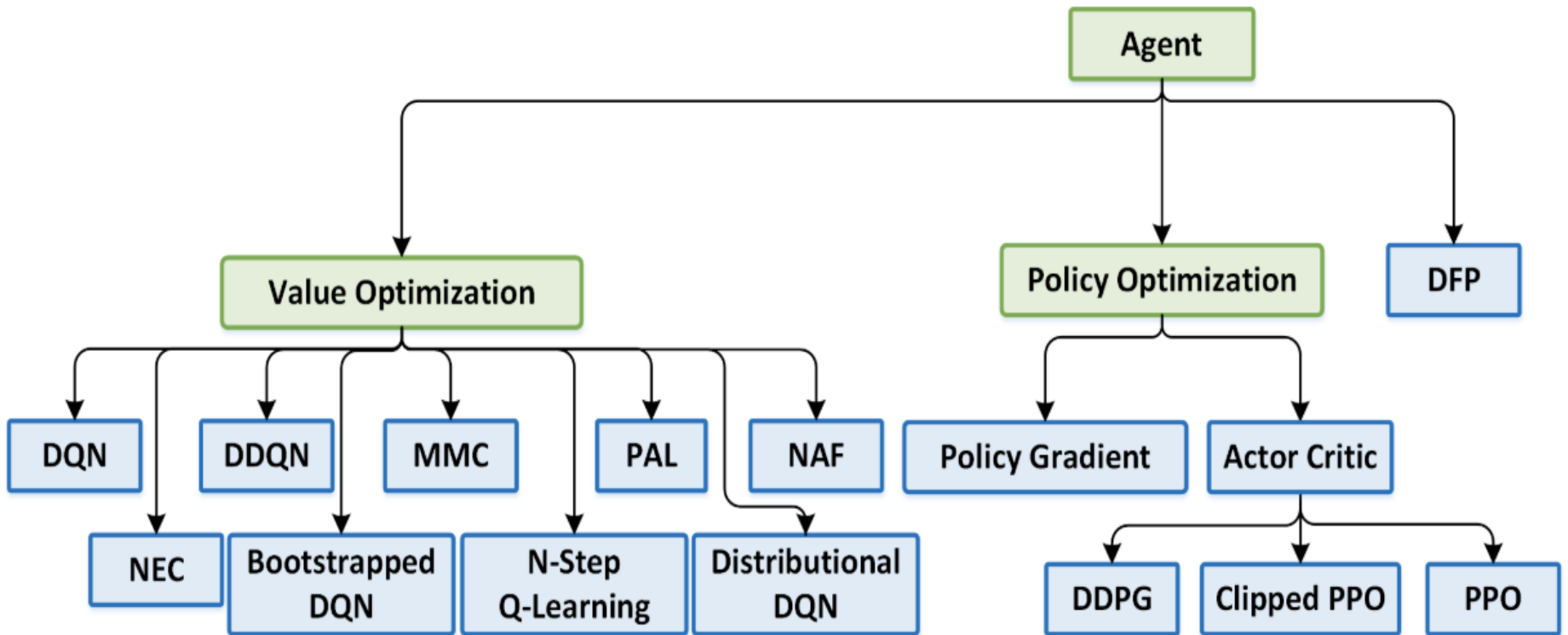
[from David Silver's course]

# Categorizing RL Algorithms



[from David Silver's course]

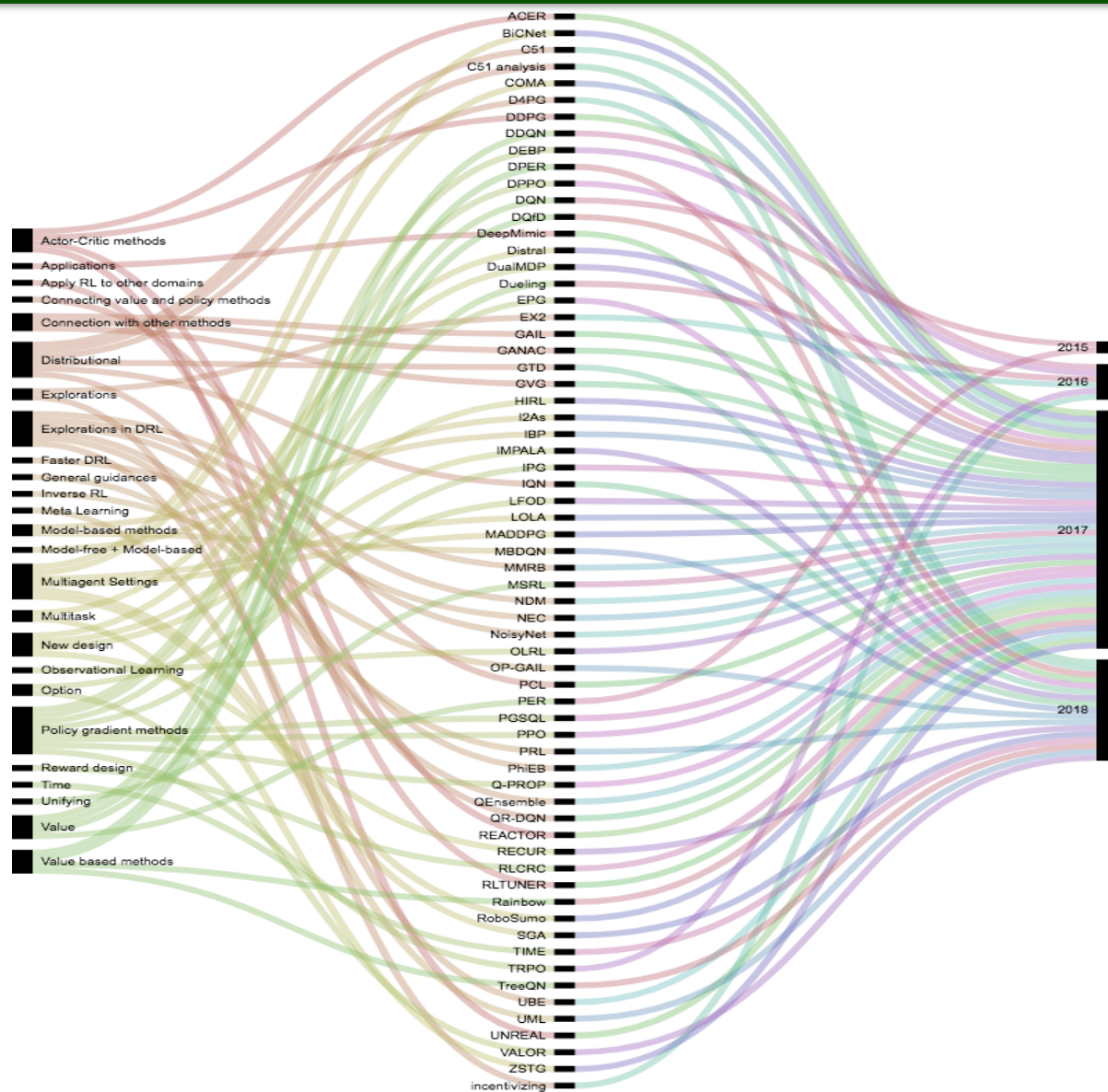
# Categorizing RL Algorithms



<https://blog.csdn.net/gsw404>



# Categorizing RL Algorithms



[from <https://blog.csdn.net/gsw404/article/details/103074046>]

## Explore, Exploit, and Explode — The Time for Reinforcement Learning is Coming

Y Yuxi Li Dec 27, 2018 · 9 min read



Reinforcement learning (RL) has been making spectacular achievements, e.g., [Atari games](#), [AlphaGo](#), [AlphaGo Zero](#), [AlphaZero](#), [AlphaStar](#), [DeepStack](#), [Libratus](#), [Catch The Flag](#), [OpenAI Five](#), [Dactyl](#), [legged robots](#), [DeepMimic](#), [learning to dress](#), [data center cooling](#), [chemical syntheses](#), [drug design](#), etc. See more [RL applications](#).

Most of these are academic research. However, we are also witnessing RL products and services, e.g., [Google Cloud AutoML](#) and [Facebook Horizon](#), and open-sources/testbeds like [OpenAI Gym](#), [Deepmind Lab](#), [Deemind Control Suite](#), [Google Dopamine](#), [Deepmind TRFL](#), [Facebook ELF](#), [Microsoft TextWorld](#), [Amazon AWS DeepRacer](#), [Intel RL Coach](#), etc. [Multi-armed bandits](#), in particular, [contextual bandits](#), have many successful applications. There are also [applications in e-commerce/recommender systems](#).

In the following, I will introduce RL briefly, discuss recent achievements, issues, research directions, applications, and the future of RL. The take-home message is: **The time for reinforcement learning is coming.**