# 强化学习前沿理论与技术

M602008B 2020 Fall

林友芳　吕凯

# 任课教师

**林友芳**　教授
学校网信办、信息中心主任

- 电话、微信：13641263964
- E-Mail：yflin@bjtu.edu.cn

吕凯 博士

- 微信：13717571377
- E-Mail：lvkai@bjtu.edu.cn

# 林友芳-简介

- **教学称号**
  - **校优秀主讲教师、最敬爱的老师**
  - **校课堂教学教风标兵、校教学名师**
  - **北京市优秀教师**
- **任课课程**
  - **《数据库系统原理》—本科课程**
  - **《数据仓库与大数据工程》—硕士课程负责人**
  - **《大数据技术》—课程负责人**
  - **《强化学习前沿理论与技术》—博士课程**

# 其他主要任职

| | |
|---|---|
| 北京交通大学网络科学与智能系统研究所 | 所长 |
| 计算机科学与技术学科 | 主要负责人 |
| --数据与知识工程方向 | 负责人 |
| --人工智能及应用 | |
| --机器学习与认知计算 | |
| 交通数据分析与挖掘北京市重点实验室 | 常务副主任 |
| 北京交通大学大数据研究院 | 院长 |
| 综合交通大数据应用技术交通行业重点实验室 | 主任 |
| 民航旅客服务智能化技术重点实验室 | 副主任 |
| 新一代信息技术及应用北京市高精尖学科 | 负责人 |

# 主要研究领域

- **大数据技术、数据挖掘与机器学习、智能技术与系统**
- **行业大数应用技术**
  - **民航、道路交通、超大型数据中心、移动通信**
- **人工智能及应用、智能系统**
  - **深度强化学习**
  - **特种交通领域CPS**
  - **JZJ自动驾驶**

# Assessment

- Explain and report on relevant frontier literature in this field. Journal and conference preferred: TPAMI, AAAI, AI, CVPR, JMLR, IJCAI, AAMAS, TCYB, JAIR, NIPS, ICML, TR, ICRA, ICLR, etc.
- Sort out and summarize recent research progress of a particular issue.
- Re-implement and explain the paper code.
- Attendance matters!

# Course Overview

- **Class hours:** 32 hours
  - 1 – 16 hours: introducing basic knowledge of reinforcement learning and extending the frontier research direction
  - 17 – 32 hours: reading and discussion of cutting-edge research topics about RL promoted by students.

- **Class objectives:**

✓ Understand Reinforcement Learning in both theoretical and applied contexts.

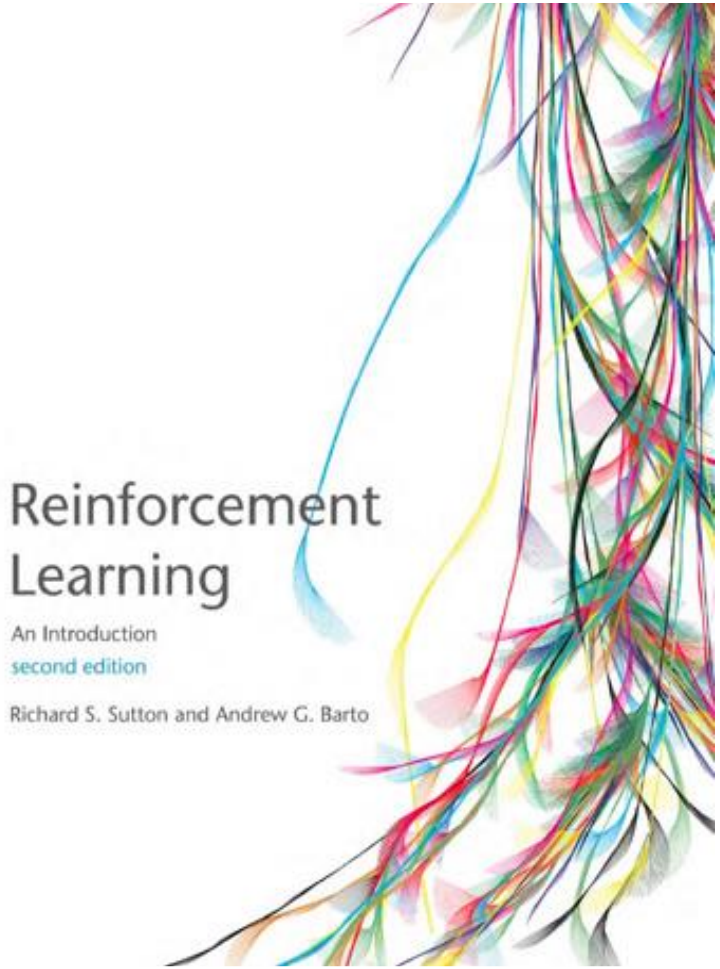
✓ Establish a fundamental theoretical structure of reinforcement learning, understand the current situation of frontier research, in order to advance the depth of theoretical research rapidly.

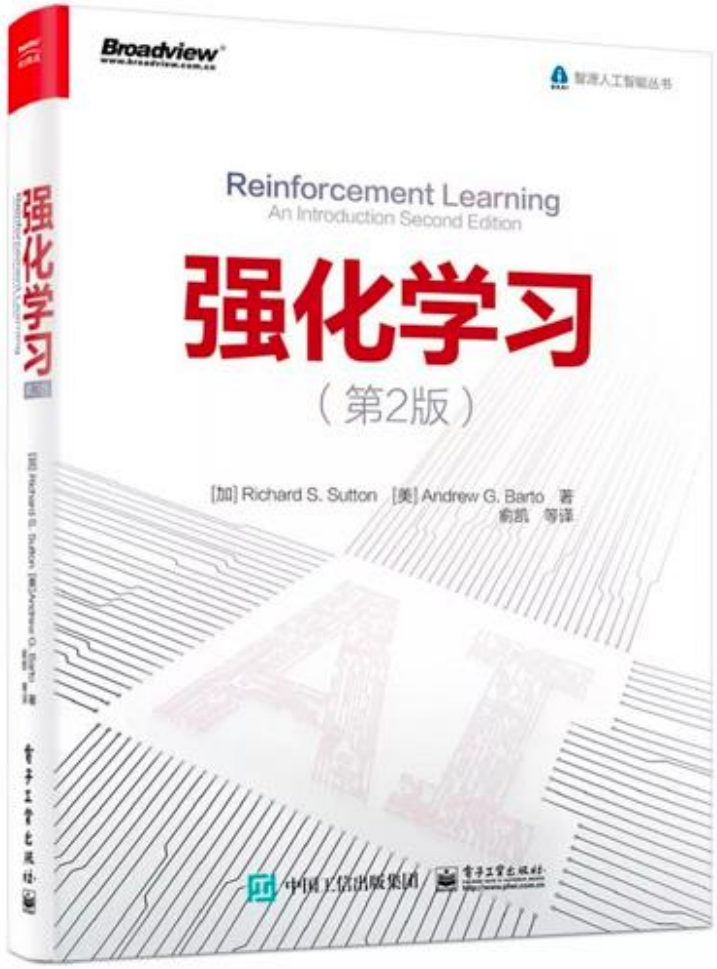
✓ Able to undertake problem identification, formulation and solution.

✓ Ability of independent critical thought, rational inquiry and self-learning.

✓ Profound respect for truth and intellectual integrity, and for the ethics of scholarship.

# Textbook: Sutton and Barto:




Available free online : http://incompleteideas.net/book/the-book-2nd.html

# Course Structure

- Lecture 1: Reinforcement Learning  Introduction
- Lecture 2: Markov Decision Process
- Lecture 3: Psychology and Neuroscience
- Lecture 4: Value-Based Reinforcement Learning
- Lecture 5: Policy-Based Reinforcement Learning
- Lecture 6: Exploration & Exploitation
- Lecture 7: Model-Based Reinforcement Learning
- Lecture 8: Imitation Learning
- Lecture 9-16: Advanced RL topics sharing

# Prerequisites

- You should have some background of Linear Algebra and Probability, as well as Machine Learning relevant knowledge(neural network, pattern recognition, deep learning, etc).
- Programming experience: Python; PyTorch, TensorFlow, etc.

# Lecture 1
# Reinforcement Learning  Introduction

# Outline

- Wisdom, Decision making, Information, Knowledge
- What is Reinforcement Learning
- What is Deep Reinforcement Learning
- Reinforcement Learning  basic components
- Reinforcement Learning relevant algorithms

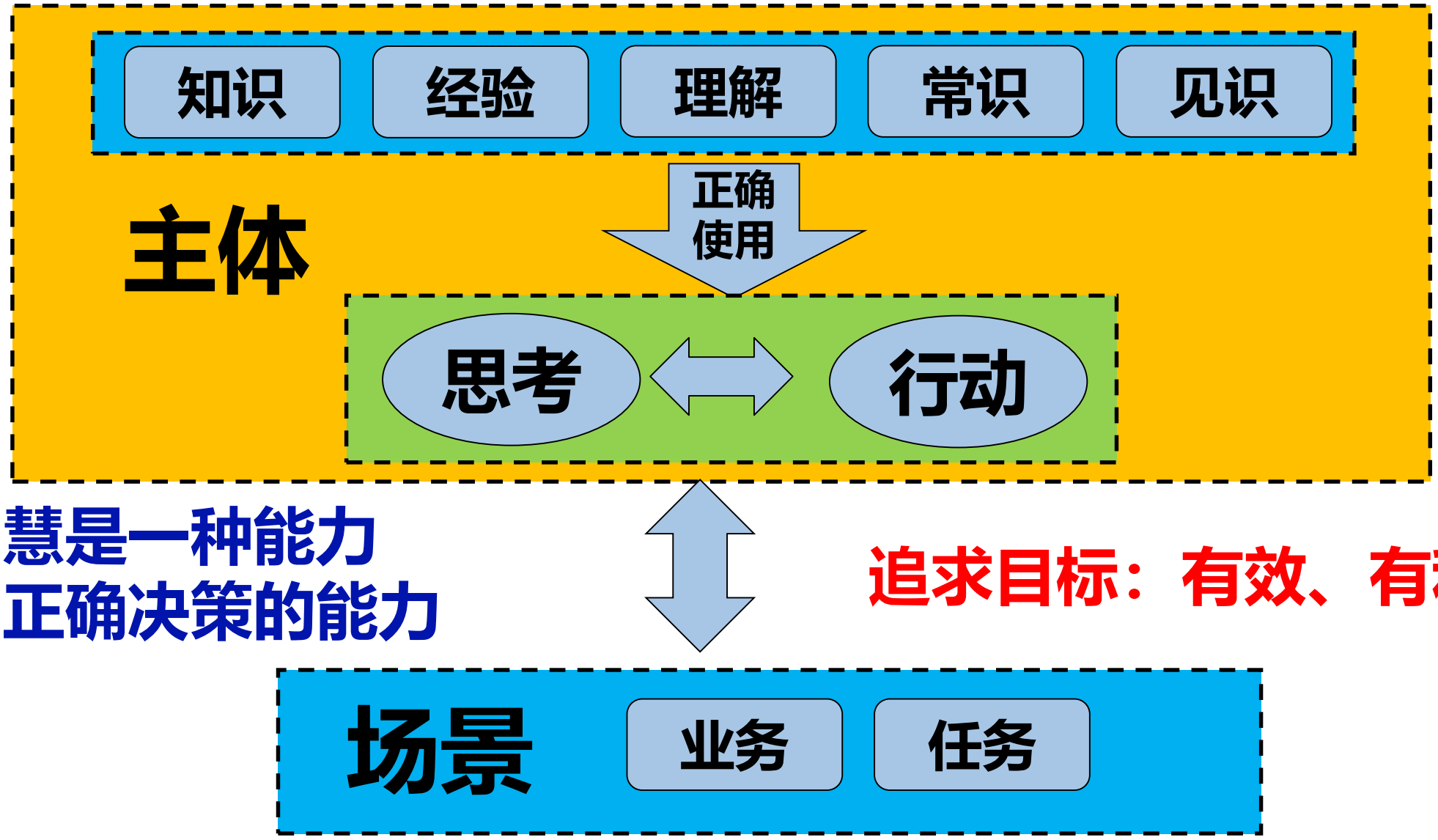
# Intelligence

- **Intelligence has been defined in many ways: the capacity for abstraction, logic, understanding, self-awareness, learning, emotional knowledge, reasoning, planning, creativity, critical thinking, and problem-solving.**
- **More generally, it can be described as the ability to perceive or infer information, and to retain it as knowledge to be applied towards adaptive behaviors within an environment or context.**

# 智慧—Wisdom

- **The ability to think and act using knowledge, experience, understanding, common sense, and insight, especially in a mature or utilitarian manner.**
- **The right use of knowledge, capacity of judging rightly in matters relating to life and conduct**
- **Displaying sound judgement in a complex, dynamic environment is a hallmark of wisdom**

# 智慧的语义分析—智慧是一种能力



# 业务场景中的决策

- **决策—Decision making**
  - **指个人、集体或自动系统为解决某个问题，借助一定的手段和方法，从若干备选方案中选择或综合成一个满意合理的方案，并付诸实施的过程。**
- **决策者或决策主体 – Decision maker**
  - **在特定场景中需要进行决策的人和组织、或自动系统**
- **一般决策模式**
  - **感知状态、利用知识、根据策略、做出选择、实施行动**

# Limited or Bounded Rationality—有限理性

- **A Primer on Decision Making—How Decisions Happen, James G. March**
- ***Studies of decision making in the real world suggest that not all alternatives are known, that not all consequences are considered, and that not all preferences are evoked at the same time.***
- ***Instead of considering all alternatives, decision makers typically appear to consider only a few and to look at them sequentially rather than simultaneously.***
- ***Decision makers do not consider all consequences of their alternatives. They focus on some and ignore others.***
- ***Relevant information about consequences is not sought, and available information is often not used.***
- ***Instead of having a complete, consistent set of preferences, decision makers seem to have incomplete and inconsistent goals, not all of which are considered at the same time.***
- ***The decision rules used by real decision makers seem to differ from the ones imagined by decision theory.***
- ***Instead of considering "expected values" or "risk" as those terms are used in decision theory, they invent other criteria. Instead of calculating the "best possible" action, they search for an action that is "good enough."***

# 问题—如何去克服人类决策的有限理性

- 所有可能的选项不是未知的
- 决策后有什么后果？产生后果的可能性？
- 思考深度问题如何克服
- 价值导向问题：个人的偏好、集体偏好
- 环境状态信息如何收集、如何表示
- 信息状态如何映射成决策：决策函数、决策规则、..
- 如何能实现尽可能的好,而不仅仅是足够好

# 信息—Information

- 信息论奠基人**香农**（Shannon）：信息是用来**消除随机不定性**的东西
- 意大利学者朗高在《信息论：新的趋势与未决问题》中认为信息是反映事物的**形成、关系和差别**的东西，它包含于事物的**差异**之中，而不在事物本身。
- 信息是物质存在的一种方式、形态或运动形态，也是事物的一种普遍属性，一般指数据、消息中所**包含的意义**，可以使消息中所描述**事件中的不定性减少**。

# Information reduces uncertainty.

- The amount of information we get grows linearly with the amount by which it **reduces our uncertainty** until the moment that **we have received all possible information** and the **amount of uncertainty is zero.**

# 关于信息与学习

- 在学习算法的学习过程中,算法存在哪些疑惑,哪些东西是信息?
- 用什么东西去表达或度量信息?
- 如何利用信息?
- 如何产生一些迷惑对手或智能体的信息?
- 如何认别出有用的信息?

# Knowledge as Justified True Belief—JTB Rules

- **The Tripartite Analysis of Knowledge:**

*S* **knows** that *p* iff

  - *p* is true;
  - *S* believes that *p*;
  - *S* is justified in believing that *p*.

- *S*是一个**主体**，*p*是一个**命题**
- *p*要**成为*S*的**知识，必须要符合的条件
- 知识可看成是***p*与*S*之间的关系**

# Things about JTB

- ***p*有可能是有条件成立**
- ***S*怎么会相信*p*?**
- ***S*是怎么调整自己的认识的，从信到不信、不信到信、半信不信?**
- **如何通过验证的过程使*S*相信?**
- **如果*p*是假怎么办？或者真假性不明?**
- ***S*相信了假东西会出现什么情况?**

# 智能体

- 智能体的能力
  - 学习、感知、逻辑、理解、情感、自醒、推理、归纳、计划、决策、创新、批判性思维、问题解决
- 具有学习能力的智能体
- 具有感知能力的智能体
- 具有理解能力的智能体
- 具有推理能力的智能体
- 具有决策能力的智能体
- 具有感知、学习、决策能力的智能体
- 具有感知、学习、推理、决策能力的智能体
- …

# 人类的成长过程

- 在父母朋辈的指导帮扶在学习成长
- 课程式学习：正规、例行性教育过程
- 专门培训：有教练
- 探索中学习，失败是成功之母
- 终生学习，持续学习

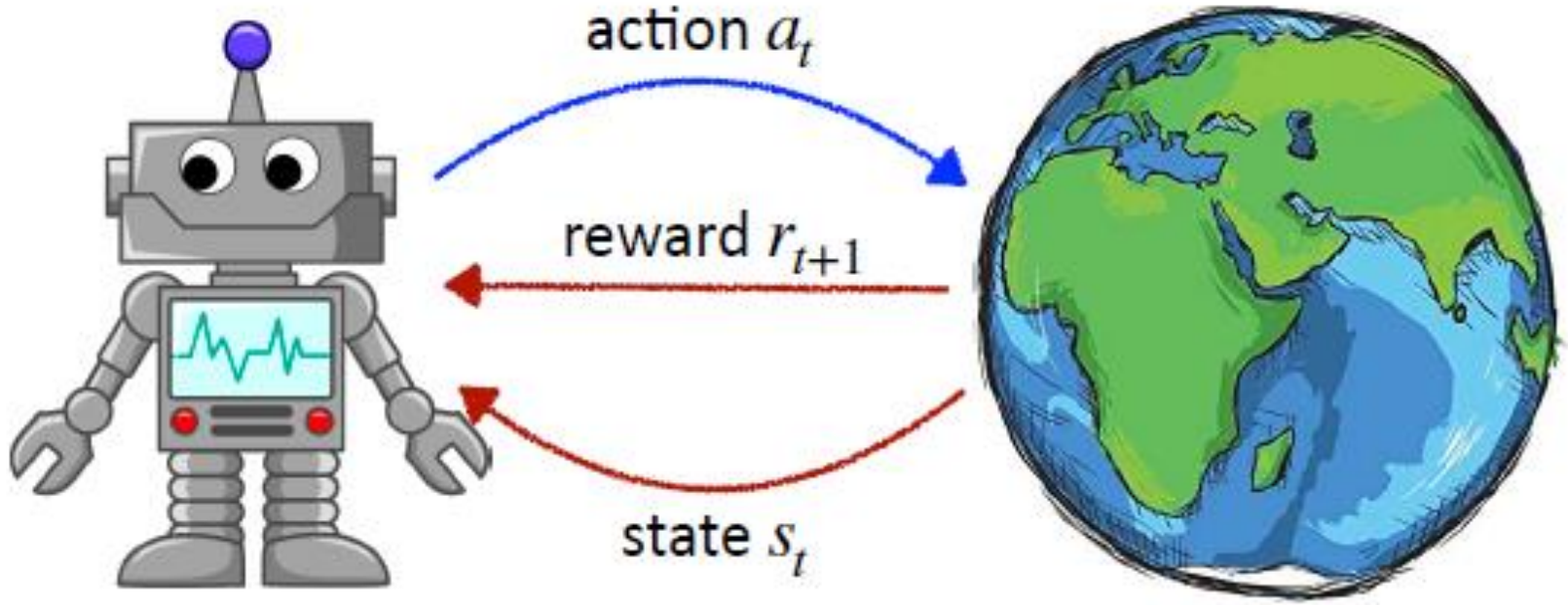
# Outline

- Wisdom, Decision making, Information, Knowledge
- What is Reinforcement Learning
- What is Deep Reinforcement Learning
- Reinforcement Learning  basic components
- Reinforcement Learning relevant algorithms

# What is Reinforcement Learning

*"A computational approach to learning whereby an agent tries to* ***maximize*** *the* ***total*** *amount of* ***reward*** *it receives while interacting with a complex and uncertain* ***environment****"*

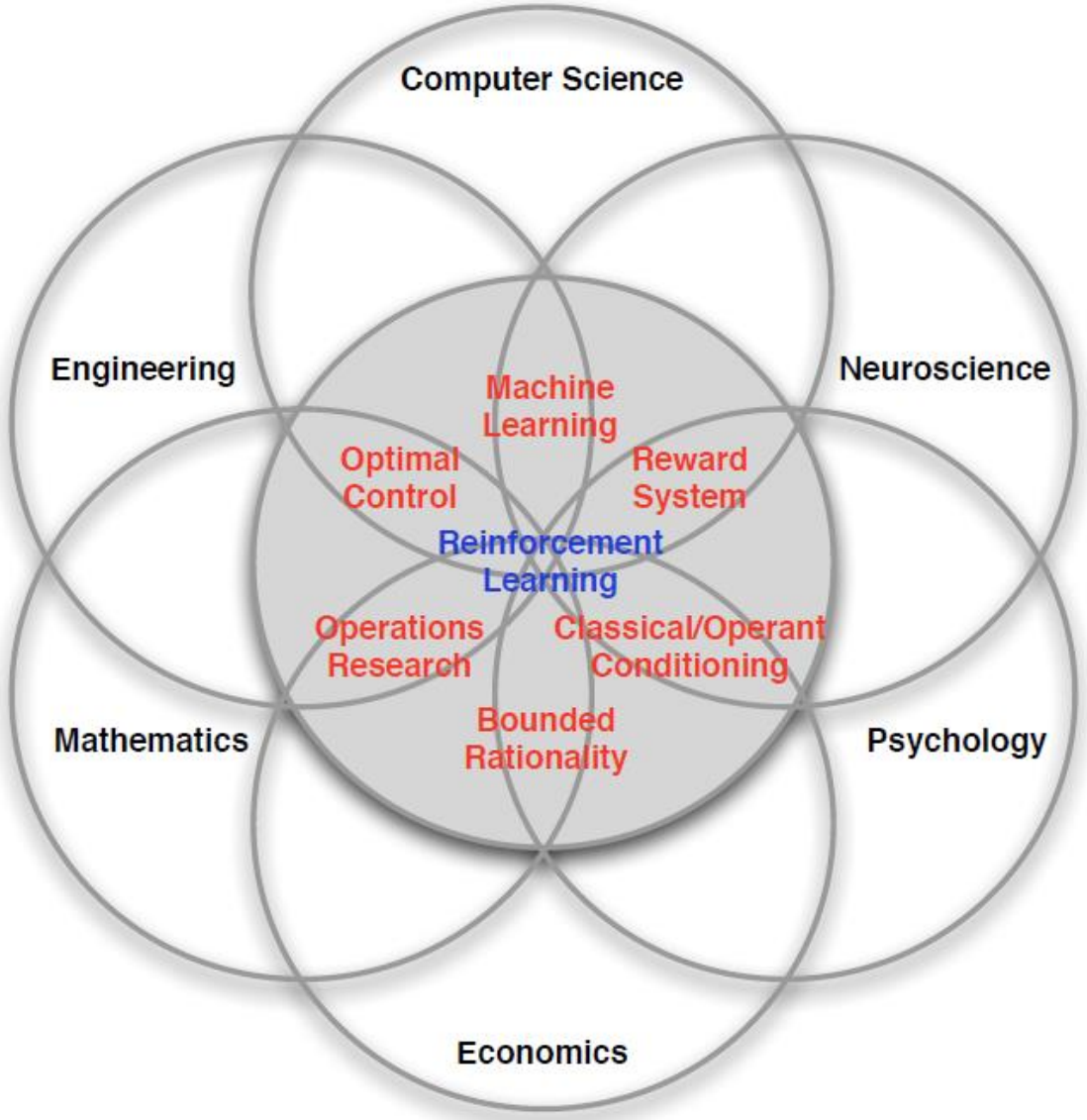
-Sutton and Barto




**The agent is to maximize the cumulative reward**

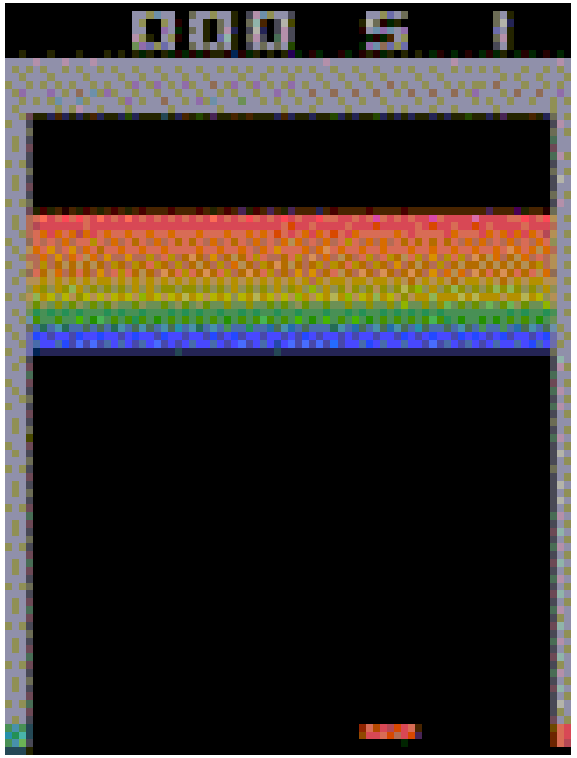
**All goals and purposes can be described by the maximization of the expected cumulative reward**

# Many Faces of Reinforcement Learning



# Games

Ping-Pong

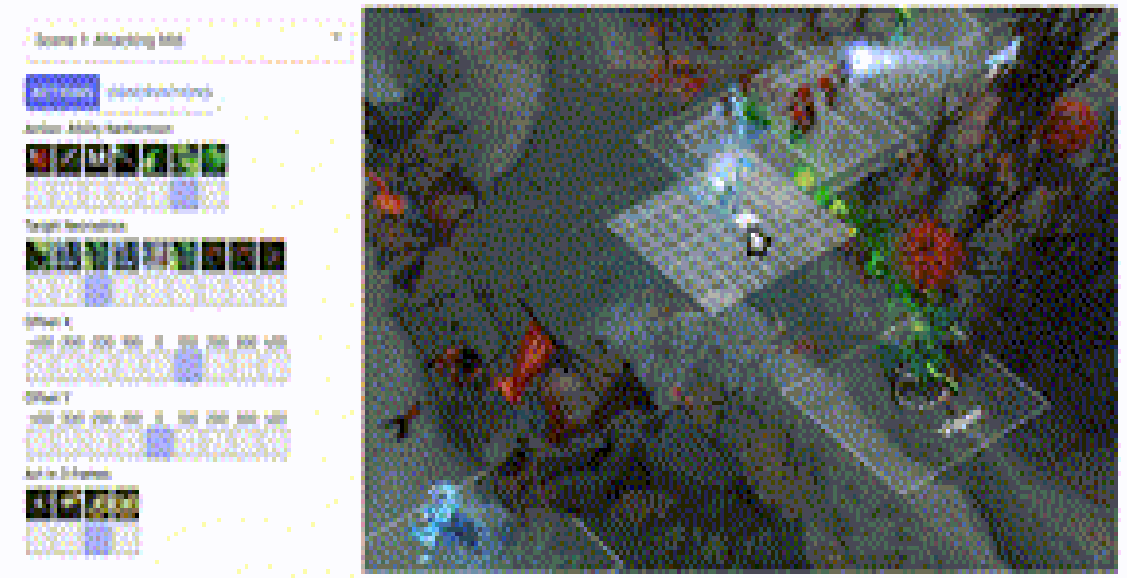



Breakout

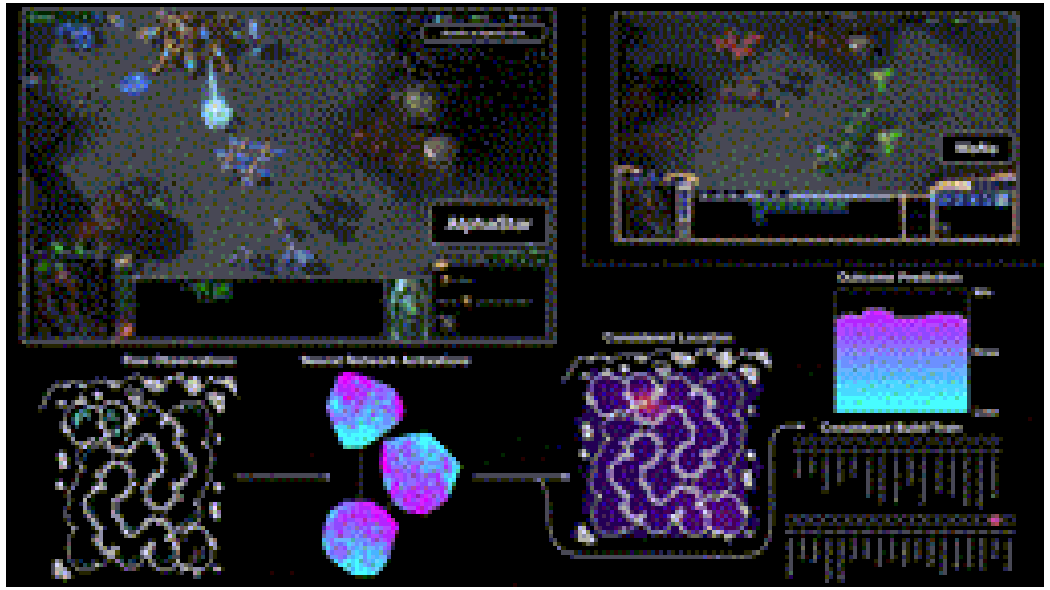

Racing

# Hide and Seek

# More recent RL problems

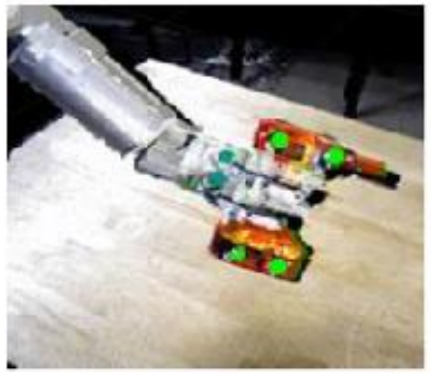


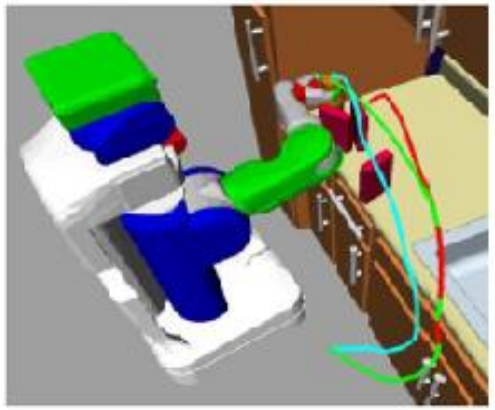
OpenAI Five playing Dota2



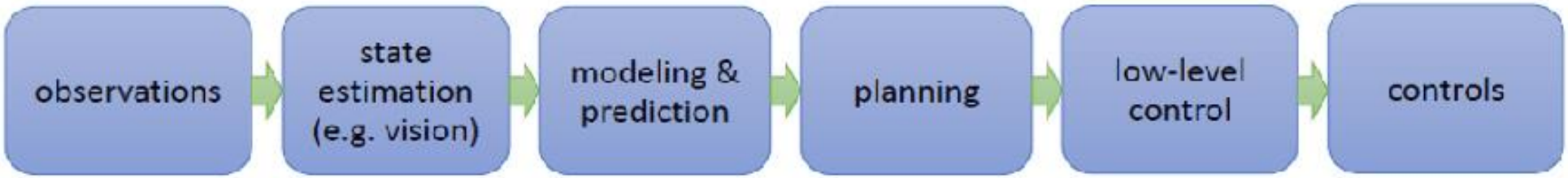
DeepMind Alpha Star playing StarCraft

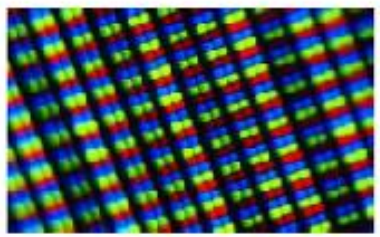
# Robotics

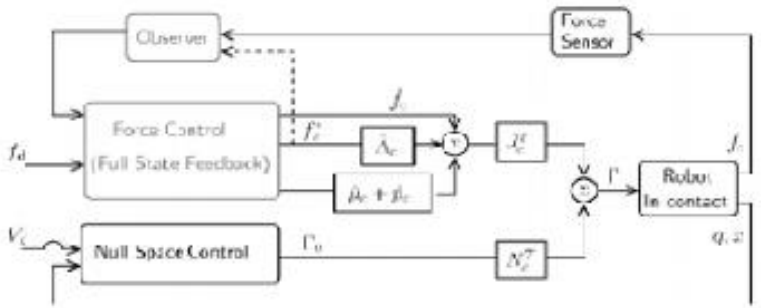




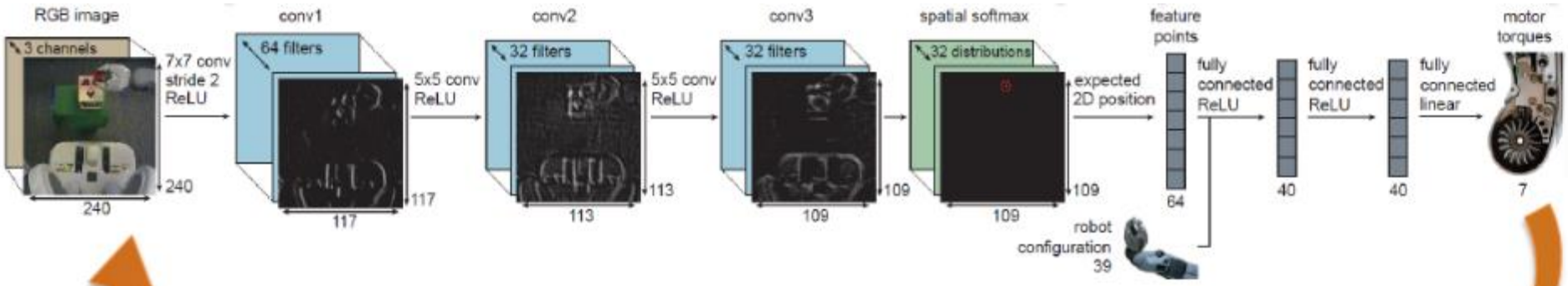
robotic control pipeline



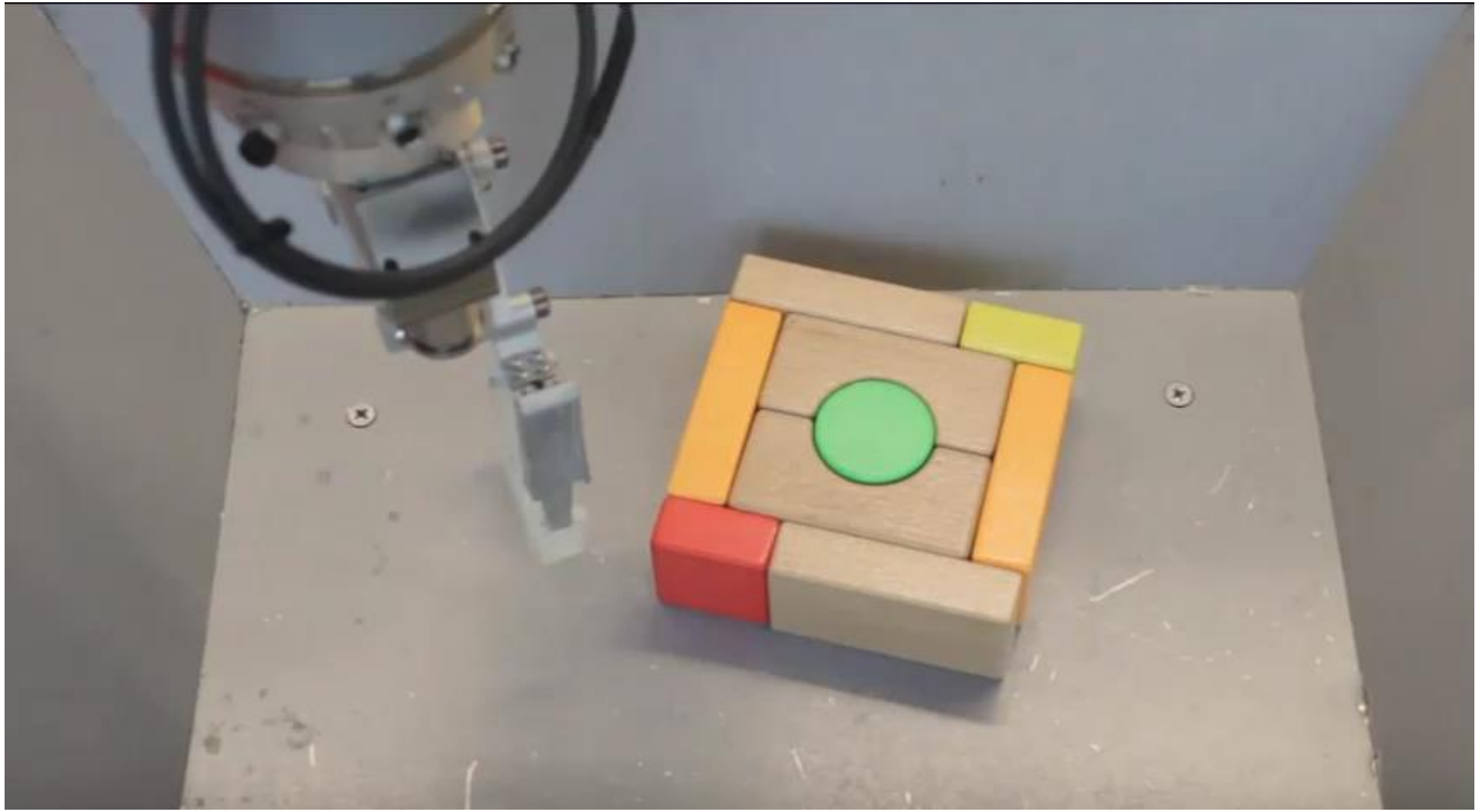


# Robotics

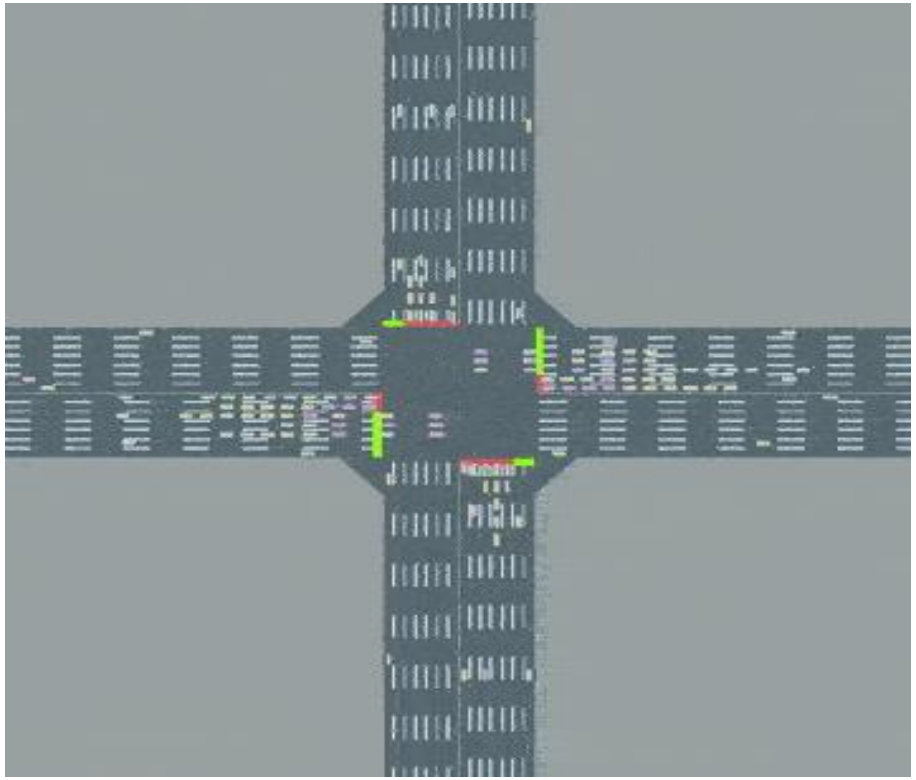



sensorimotor loop

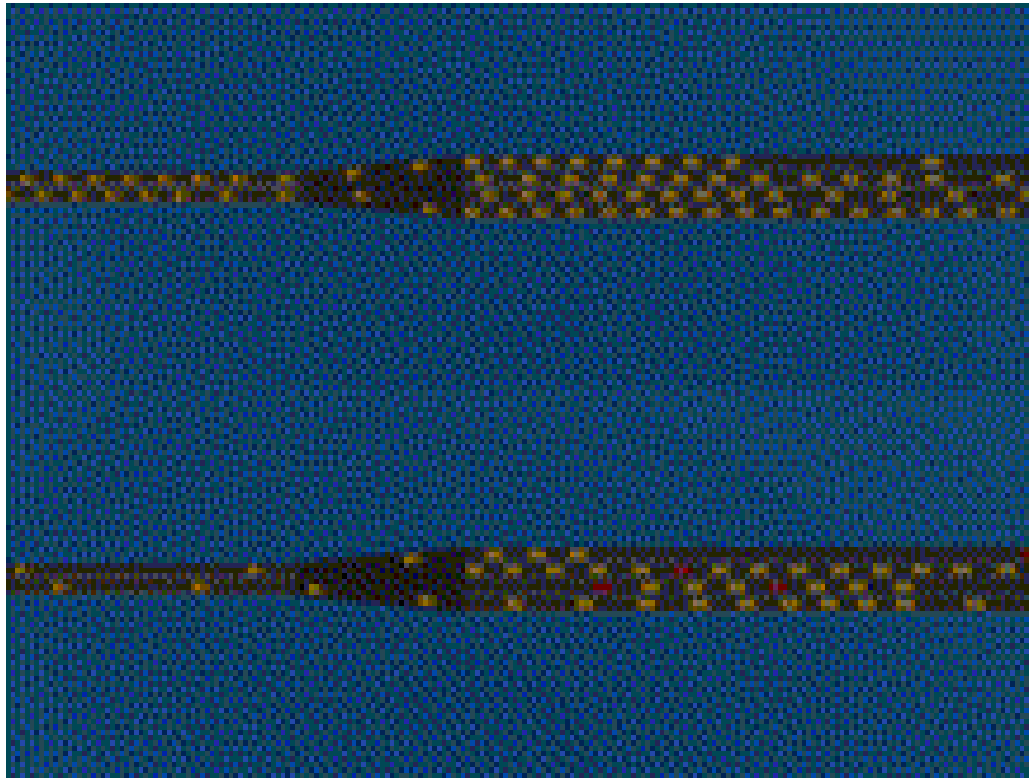
# Robotics - Learning to grasp



# Not just games and robotics!



Traffic lights control



Traffic  control

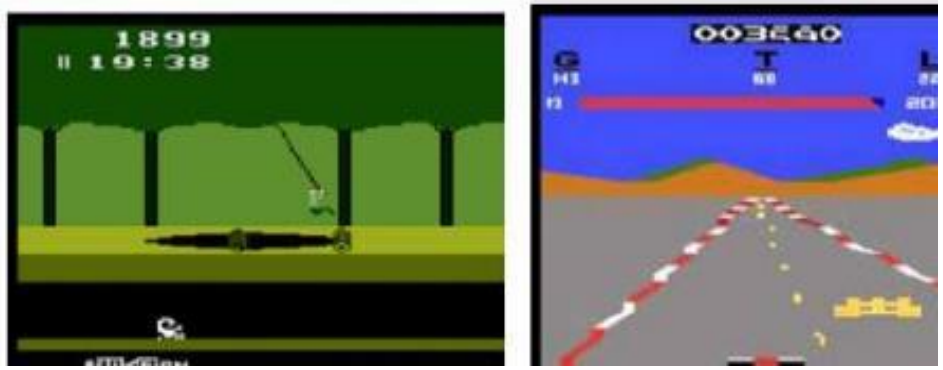
# Recommendation in E-commerce

→ prediction: this sells good → decision: recommend more → sold more

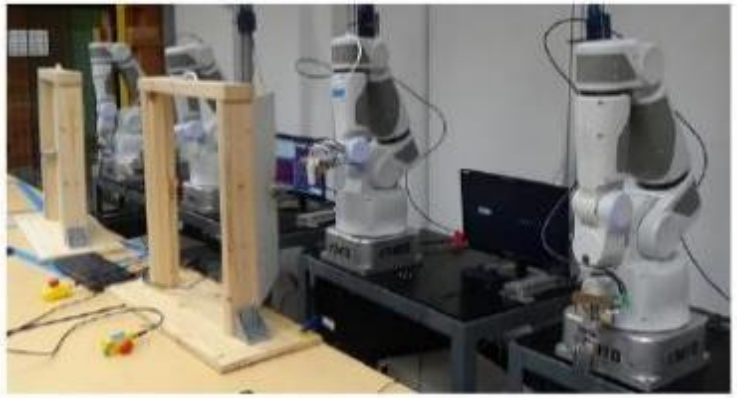
→ prediction: this looks bad → decision: recommend less → sold less

# Why RL works now?

- Computation power: many GPUs to do trial-and-error rollout.
- Acquire the high degree of proficiency in domains governed by simple, known rules.
- End-to-end training, features and policy are **jointly optimized** toward the end goal.




Game Playing



Robotics


Beating best human player

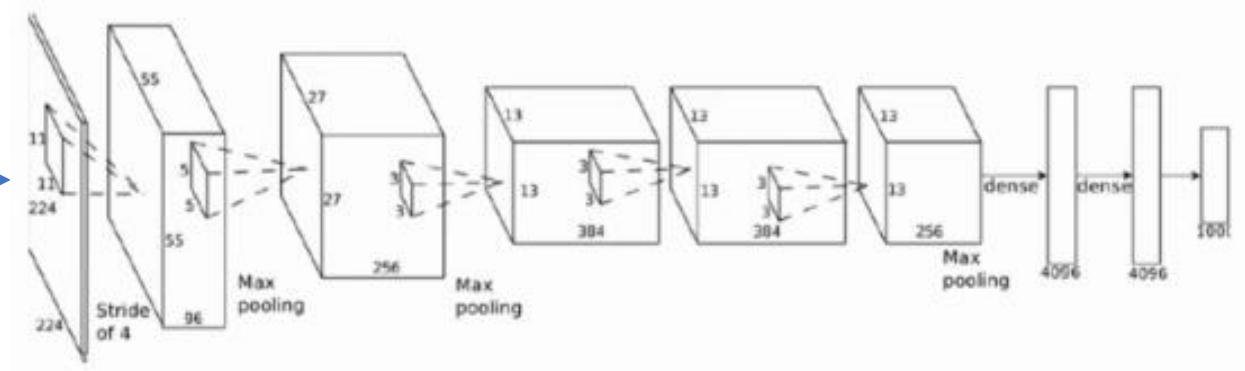
# Outline

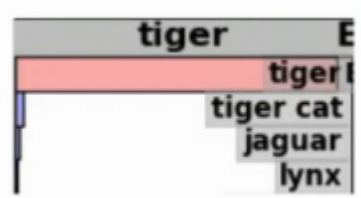
- Wisdom, Decision making, Information, Knowledge
- What is Reinforcement Learning
- What is Deep Reinforcement Learning
- Reinforcement Learning  basic components
- Reinforcement Learning relevant algorithms

# Deep Learning Task

Instance

x

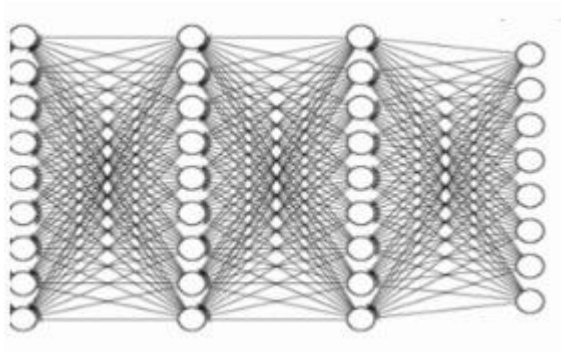


# Reinforcement Learning Task

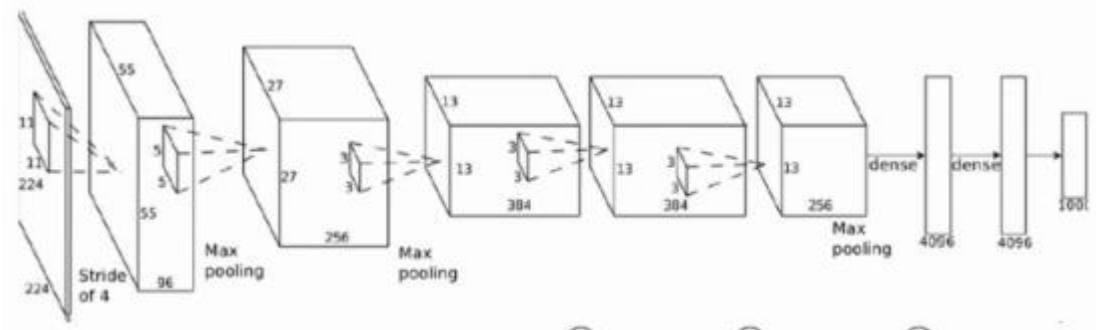
Instance

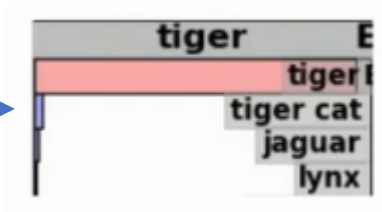
x

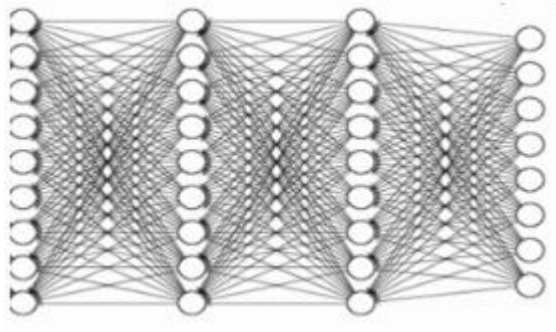


Run away!

perception



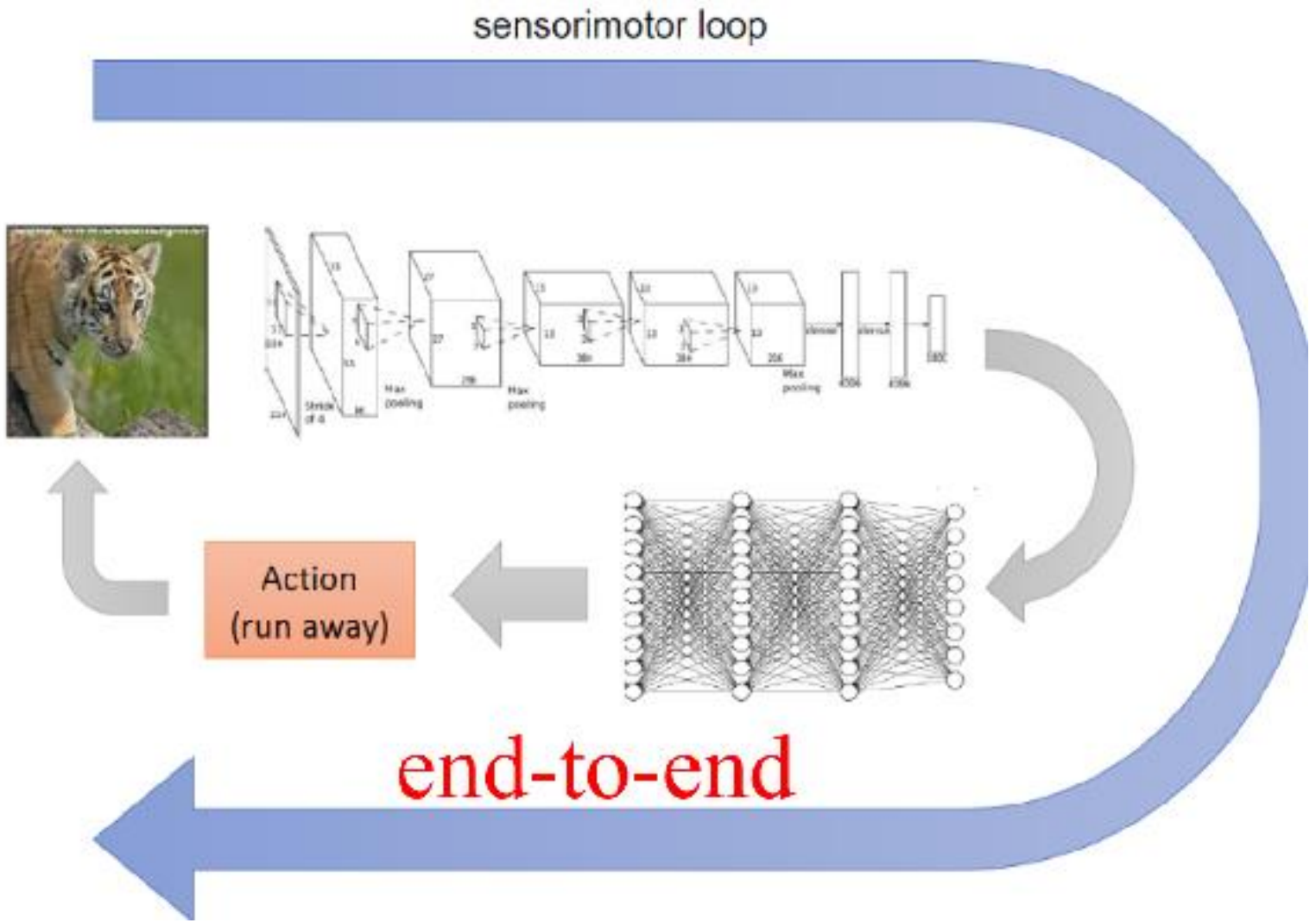



Action
(run away)

action

# Deep Reinforcement Learning:
# Deep Learning + Reinforcement Learning




- DL： can process complex sensory input
- RL： can choose complex actions

# Characteristics of Reinforcement Learning

*What makes reinforcement learning different from other machine learning paradigms?*

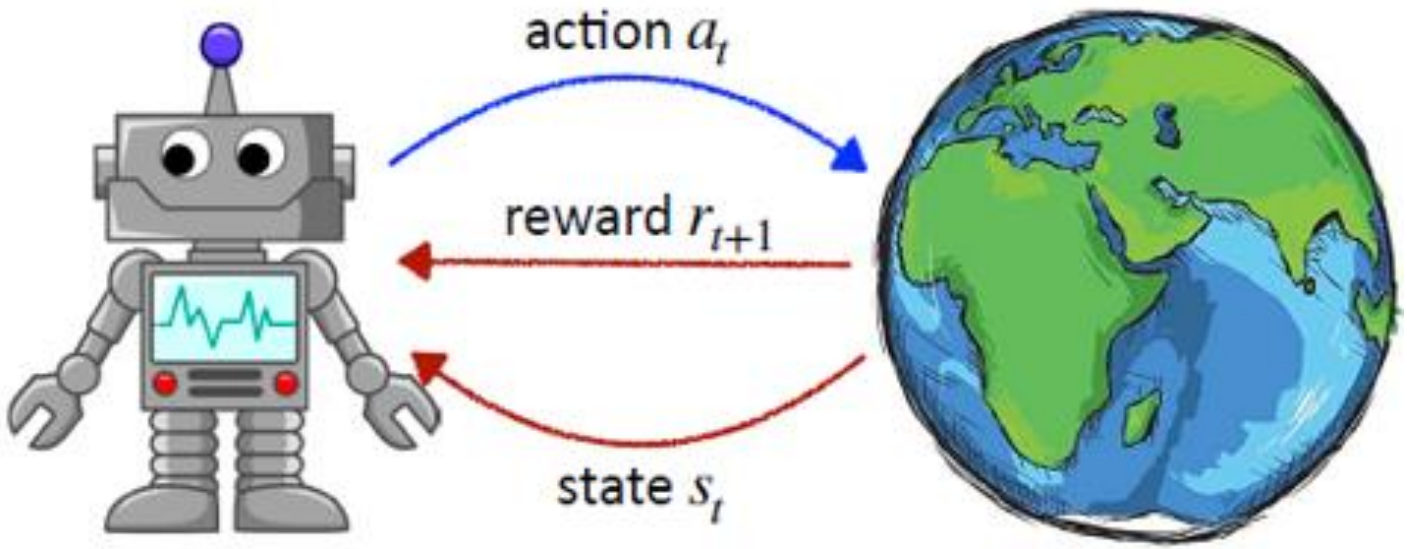
- There is no supervisor, only a reward signal, trial-and-error
- Feedback is delay, not instantaneous
- Time really matters
- Agent's action affects the subsequent data it receives

# Outline

- Wisdom, Decision making, Information, Knowledge
- What is Reinforcement Learning
- What is Deep Reinforcement Learning
- Reinforcement Learning  basic components
- Reinforcement Learning relevant algorithms

# RL components

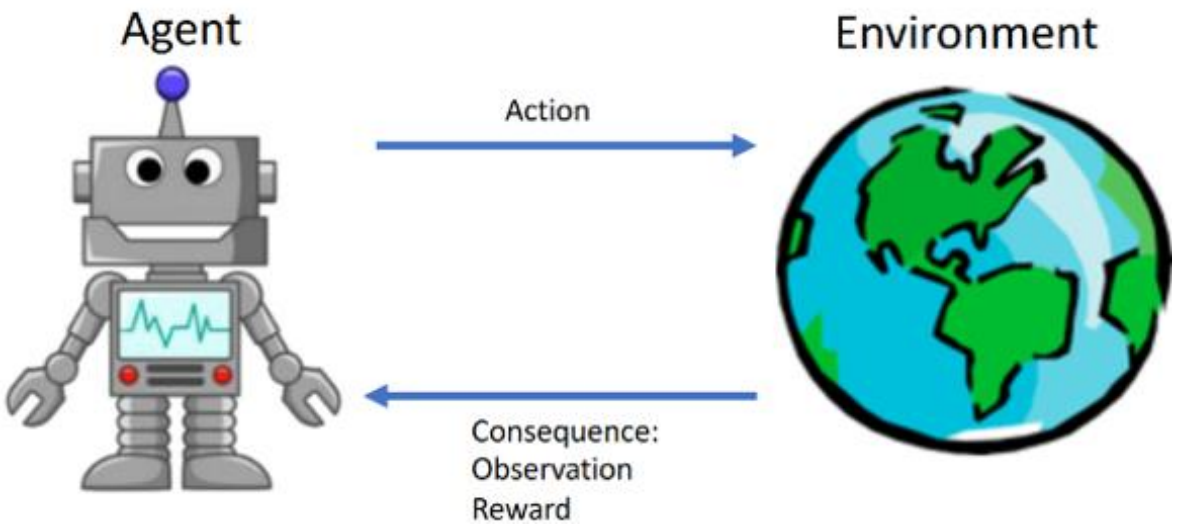
- Agent
- Environment
- State
- Reward



# Agent and Environment

Agents are the software programs that make intelligent decisions and they are basically learners in RL. Agents take action by interacting with the environment and they receive rewards based on their actions.

Everything agents interact with is called an environment. The environment is the outside world. It comprises everything outside the agent.



# Environment

- **Deterministic**

An environment is said to be deterministic when we know the outcome based on the current state.

- **Stochastic**

An environment is said to be stochastic when we cannot determine the outcome based on the current state. There will be a greater level of uncertainty. For example, we never know what number will show up when throwing a dice.

- **Fully observable**

When an agent can determine the state of the system at all times, it is called fully observable. For example, in a chess game, the state of the system, that is, the position of all the players on the chess board, is available the whole time so the player can make an optimal decision.

# Environment

- **Partially observable**

When an agent cannot determine the state of the system at all times, it is called partially observable. For example, in a poker game, we have no idea about the cards the opponent has.

- **Discrete**

When there is only a finite state of actions available for moving from one state to another, it is called a discrete environment.

- **Continuous**

When there is an infinite state of actions available for moving from one state to another, it is called a continuous environment.

# Environment

- **Episodic and non-episodic**

The episodic environment is also called the non-sequential environment. In an episodic environment, an agent's current action will not affect a future action, whereas in a non-episodic environment, an agent's current action will affect a future action and is also called the sequential environment.

- **Single and multi-agent**

A single-agent environment has only a single agent and the multi-agent environment has multiple agents. Multi-agent environments are extensively used while performing complex tasks. There will be different agents acting in completely different environments. Agents in a different environment will communicate with each other. A multi-agent environment will be mostly stochastic as it has a greater level of uncertainty.

# History and State

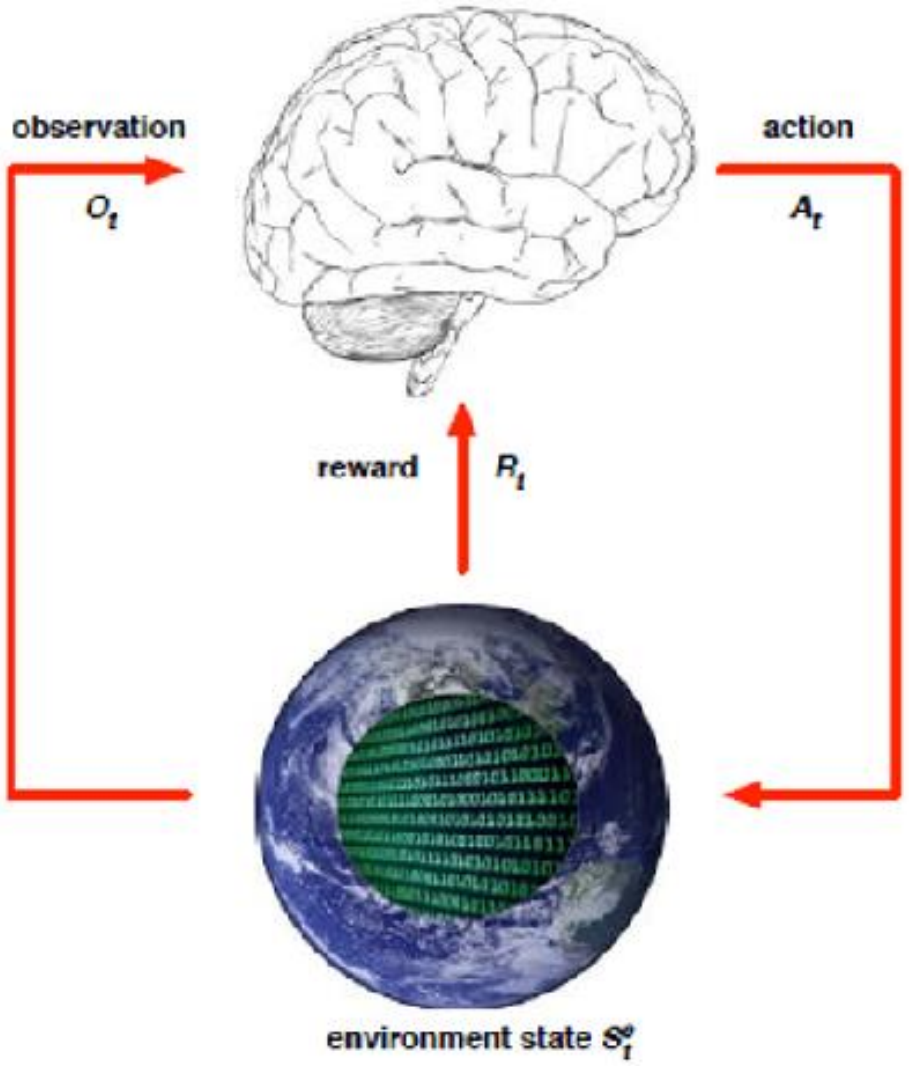
- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$$

- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- Formally, state is a function of the history:

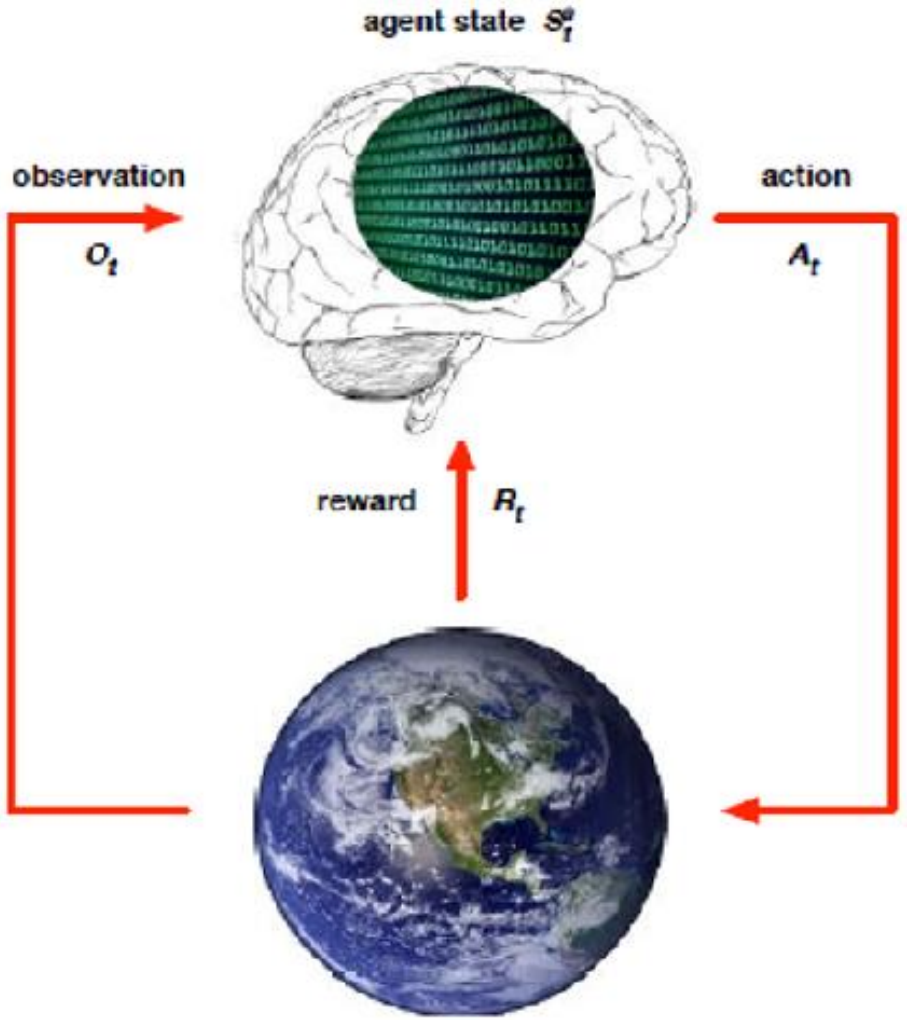
$$S_t = f(H_t)$$

# Environment State




- The environment state $S_t^e$ is the environment's private representation
- *i.e.* whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if $S_t^e$ is visible, it may contain irrelevant information

# Agent State




- The agent state $S_t^a$ is the agent's internal representation
- *i.e.* whatever information the agent uses to pick the next action
- *i.e.* it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

# Information State

- An information state (a.k.a. Markov state) contains all useful information from the history.

**Definition**

A state $S_t$ is Markov if and only if

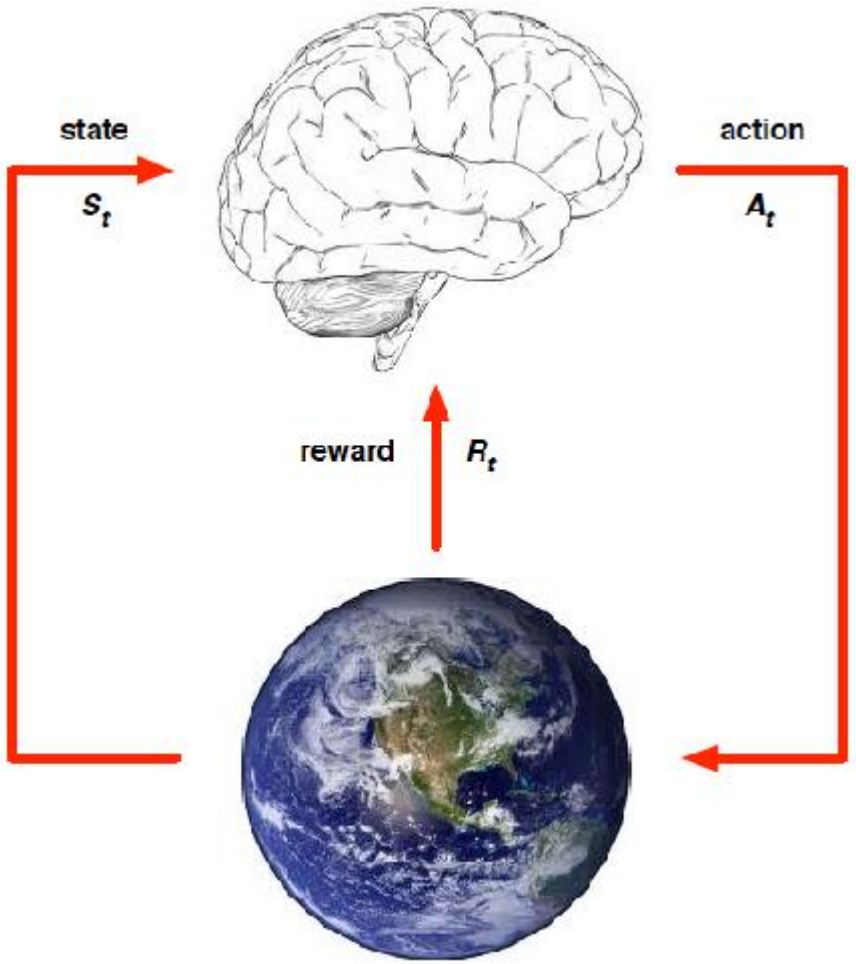
$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

- "The future is independent of the past given the present"

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

- Once the state is known, the history may be thrown away
- *i.e.* The state is a sufficient statistic of the future
- The environment state $S_t^e$ is Markov
- The history $H_t$ is Markov

# Fully Observable Environments

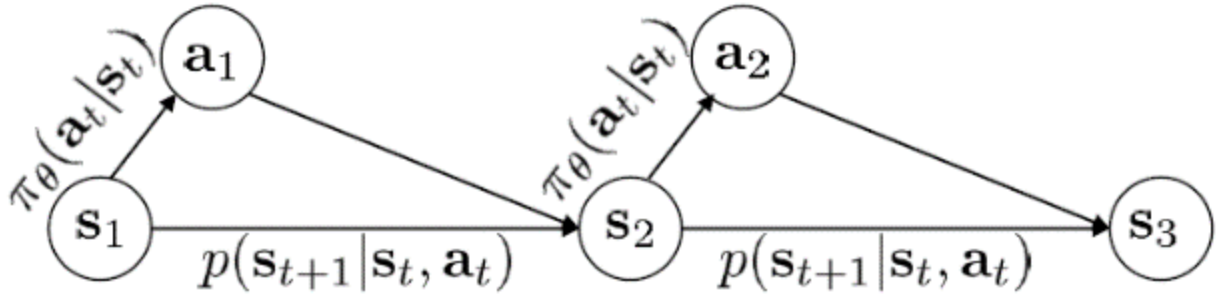



- Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

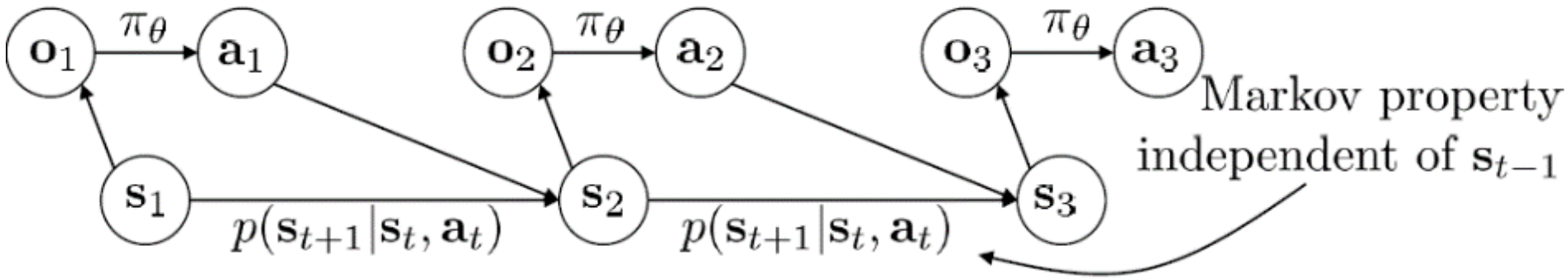
- Agent state = environment state = information state

Formally, this is a Markov decision process(MDP) – next lecture



# Partially Observable Environments

- Partial observability: agent indirectly observes environment:
  - 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
- Now agent state ≠ environment state
- Formally this is a partially observable Markov decision process (POMDP)



# Reward

- A reward is a scalar feedback signal
- Indicate how well agent is doing at time step *t*
- Reinforcement Learning is based on the maximization of rewards:

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

# Examples of Rewards

- Chess players play to win:

+/-reward for wining or losing a game

- Gazelle calf struggles to stand:

+/-reward for running with its mom or being eaten

- Manage stock investment

+/-reward for each profit or loss in $

- Play Atari games

+/-reward for increasing or decreasing scores

# Sequential Decision Making

- Objective of the agent: select a series of actions to maximize total future rewards
- Actions may have long term consequences
- Reward may be delayed
- Trade-off between immediate reward and long-term reward

Examples:

- A financial investment (may take months to mature)
- Refuelling a helicopter (might prevent a crash in several hours)
- Blocking opponent moves (might help winning chances many moves from now)

# Major Components of an RL Agent

- **Policy**:
  - agent's behaviour function
- **Value Function**:
  - how good is each state and/or action
- **Model**:
  - agent's representation of the environment

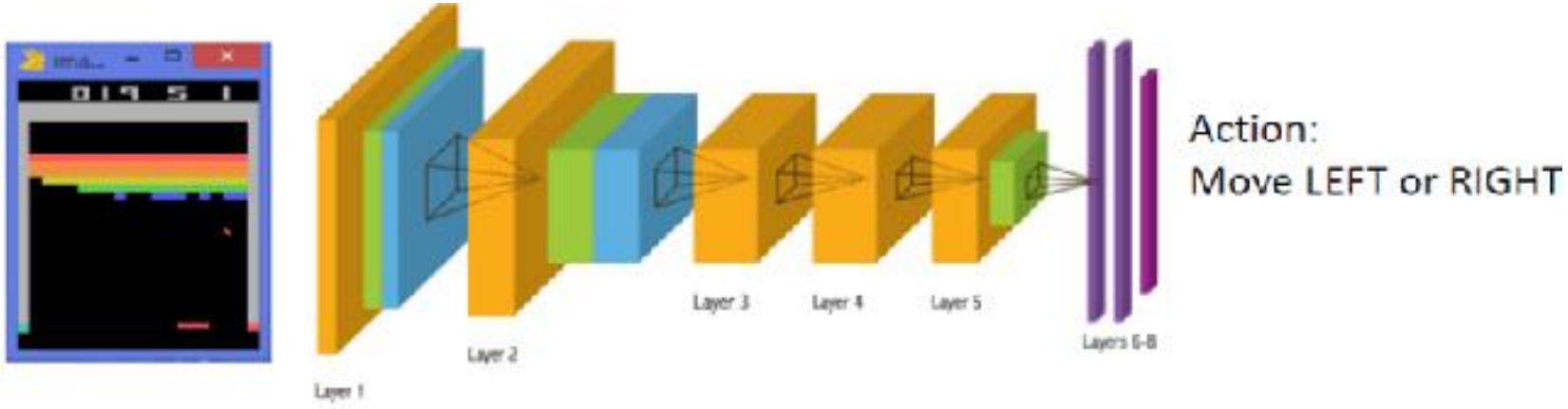
# Policy

- A policy defines the agent's behaviour in an environment. It is a **map from state to action**. A policy is often denoted by the symbol **π**. A policy can be in the form of a lookup table or a complex search process.
- Deterministic policy:

$$a = \pi(a|s)$$

- Stochastic policy:

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



# Policy

- On policy
  - "Learn on the job"
  - Learn about policy from experience sampled from $\pi$
- Off policy
  - "Look over someone's shoulder"
  - Learn about policy $\pi$ from experience sampled from $\mu$

# Value Function

- Value function: expected discounted sum of future rewards under a particular policy **π**
- Discount factor weights immediate vs future rewards
- Used to quantify goodness/badness of states and actions

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

- Q-function (could be used to select among actions)

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

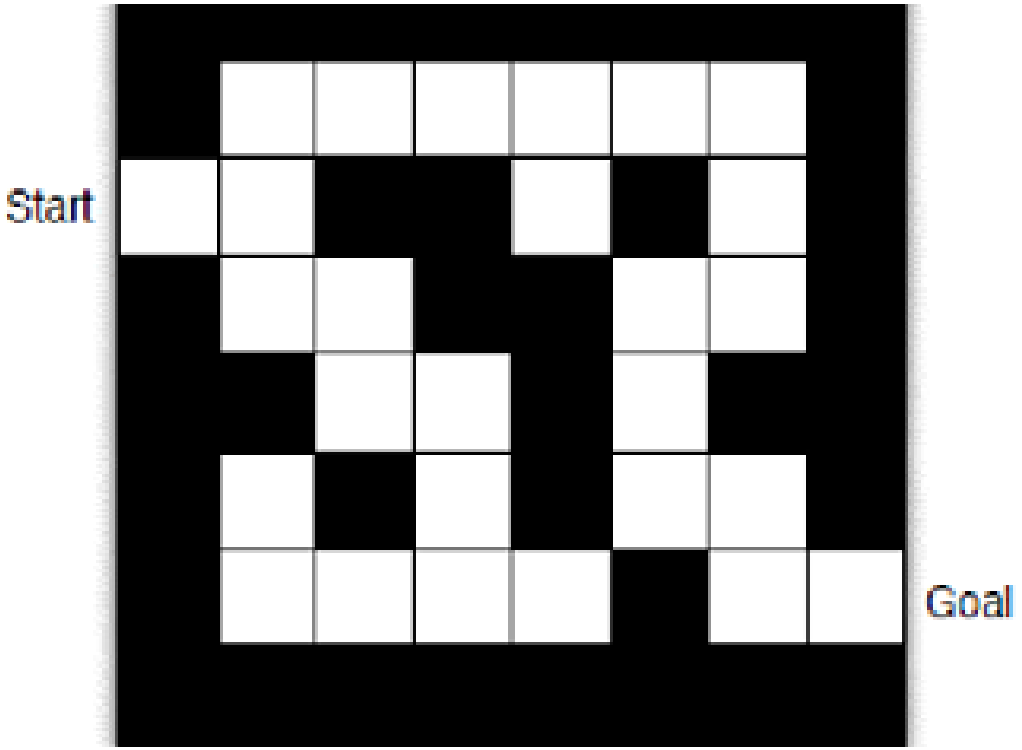
# Model

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward:

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

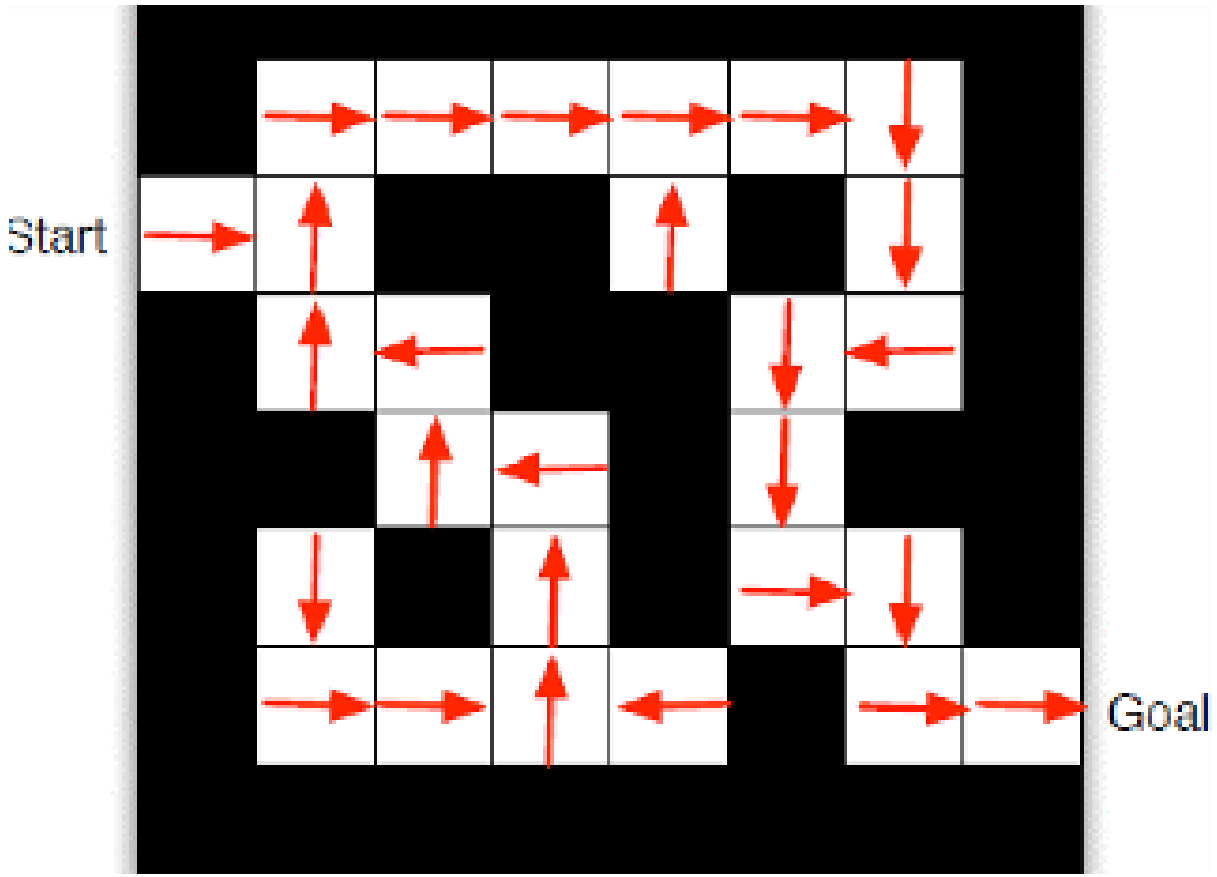
$$\mathcal{R}_{s}^{a} = \mathbb{E}[\mathcal{R}_{t+1} \mid S_t = \mathrm{s}, A_t = a]$$

# Maze Example



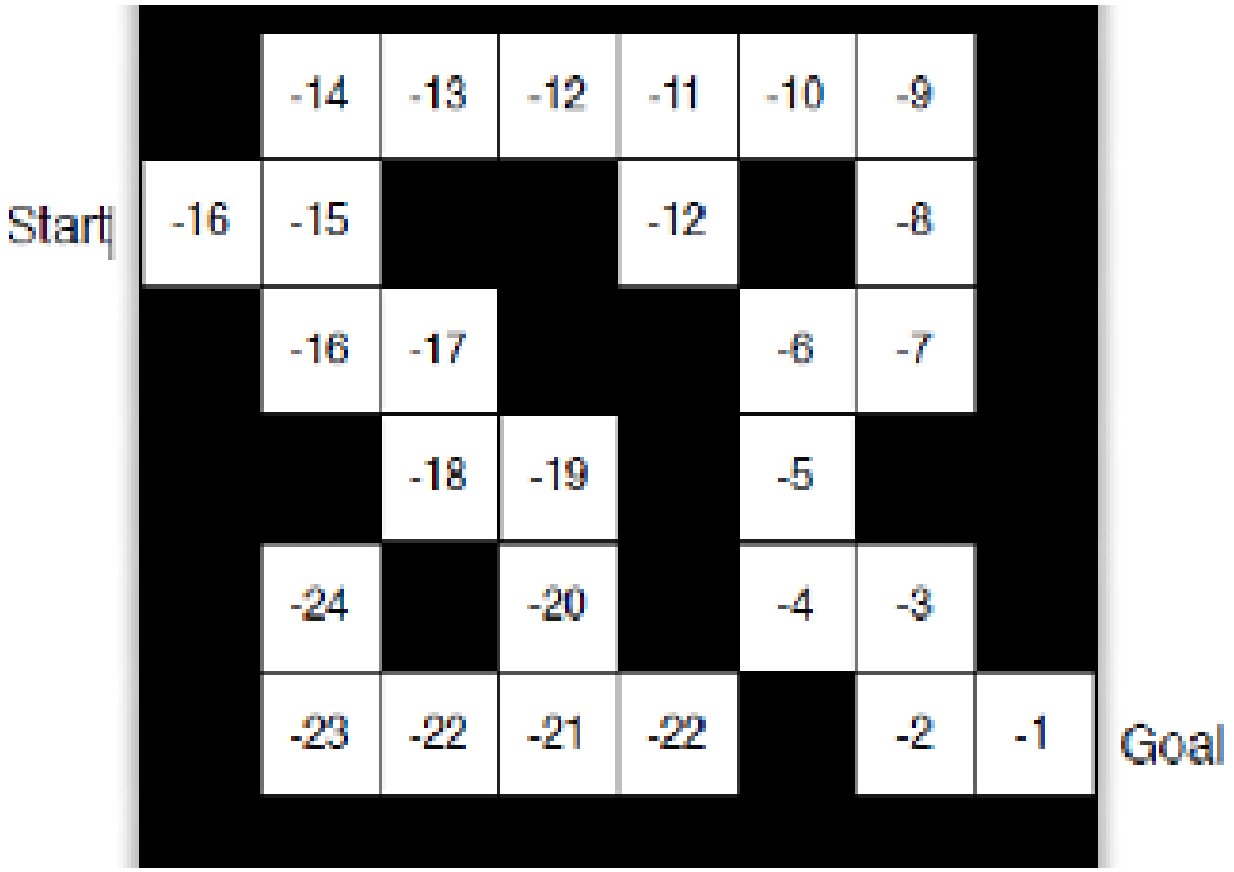

- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

# Maze Example: Policy-based RL



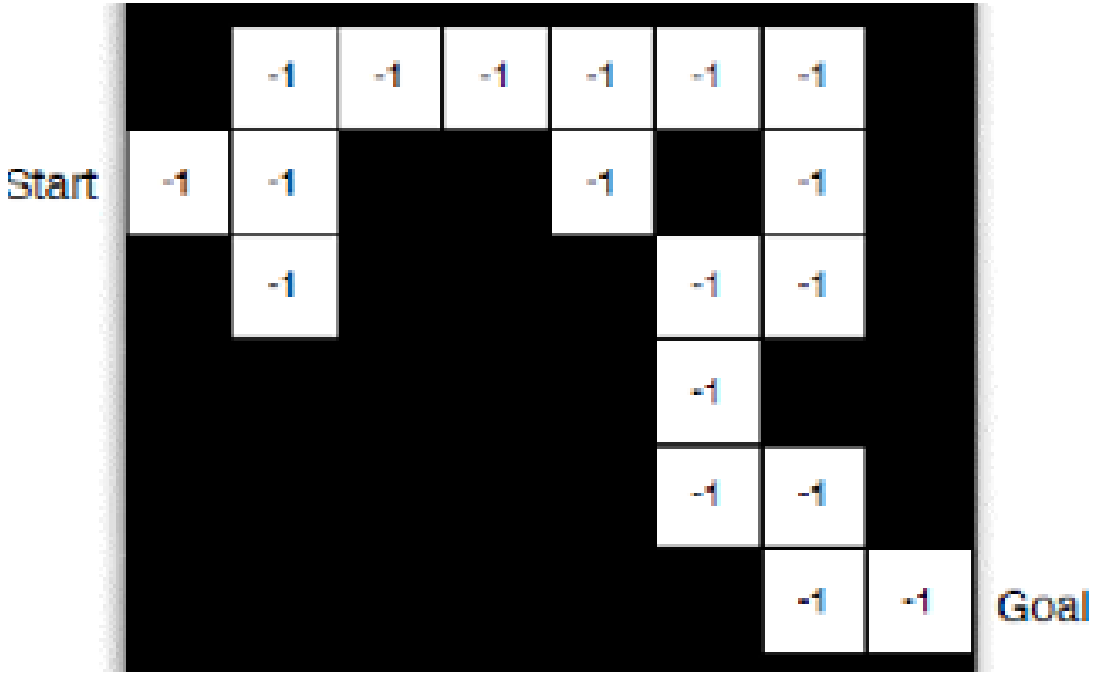

Arrows represent policy $\pi(s)$ for each state s

# Maze Example: Value-based RL




Numbers represent value $V_\pi(s)$ of each state s

# Maze Example: Model -based RL




- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $\mathcal{P}_{ss'}^{a}$,
- Numbers represent immediate reward $\mathcal{R}_{s}^{a}$ from each state s (same for all a)

# Types of RL Agents based on What the Agent Learns

- Value-based agent
  - Value Function
  - No Policy (Implicit)
- Policy-based agent:
  - Policy
  - No Value Function(Implicit)
- Actor-Critic agent:
  - Policy
  - Value Function

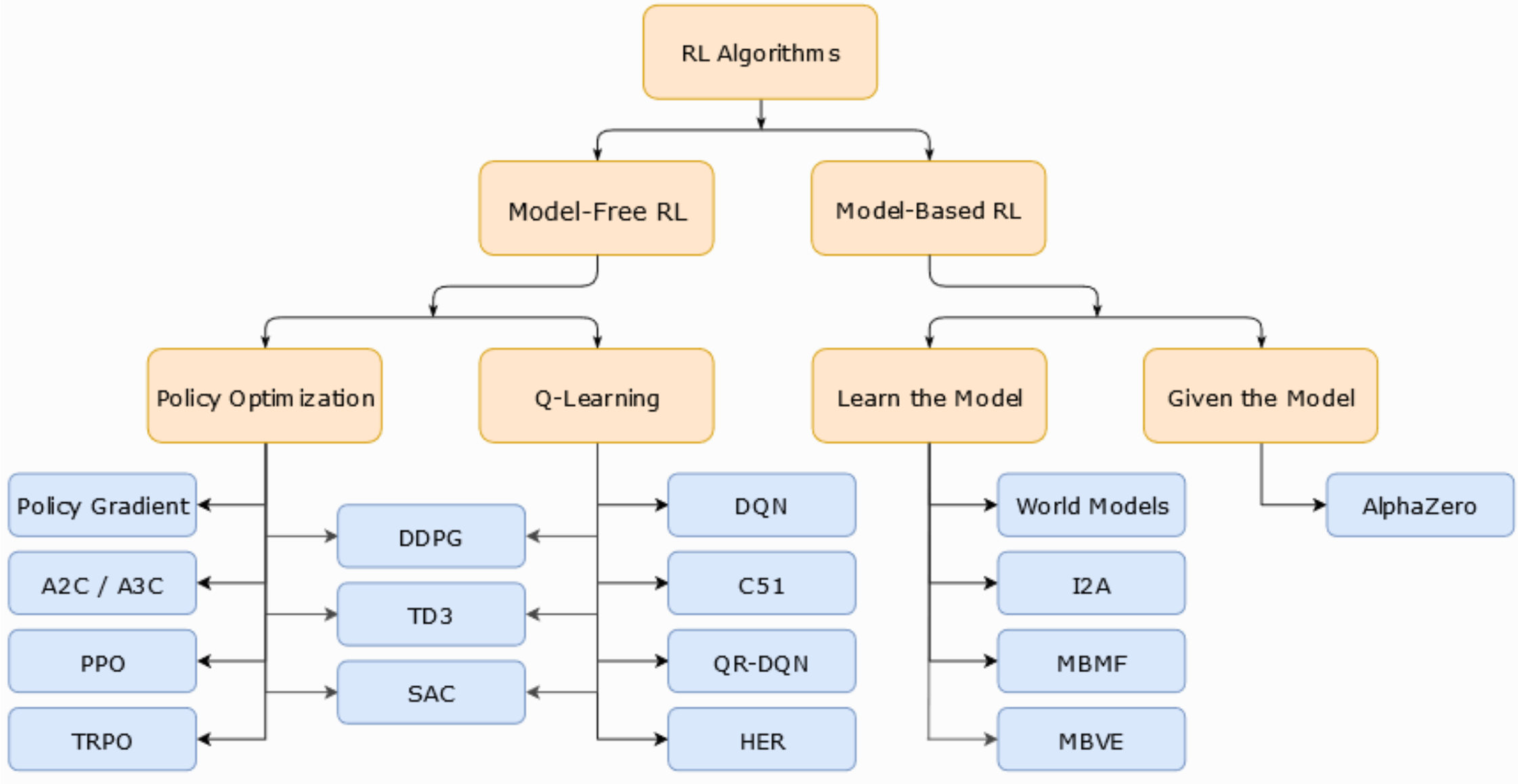
# Types of RL Agents on if there is model

- Model-based: estimate the transition model, and then…
  - Use it for planning (no explicit policy)
  - Use it to improve a policy
  - Something else
- Model-free
  - Value Function and/or Policy Function
  - No Model

# Outline

- Wisdom, Decision making, Information, Knowledge
- What is Reinforcement Learning
- What is Deep Reinforcement Learning
- Reinforcement Learning  basic components
- Reinforcement Learning relevant algorithms

# Types of RL algorithms

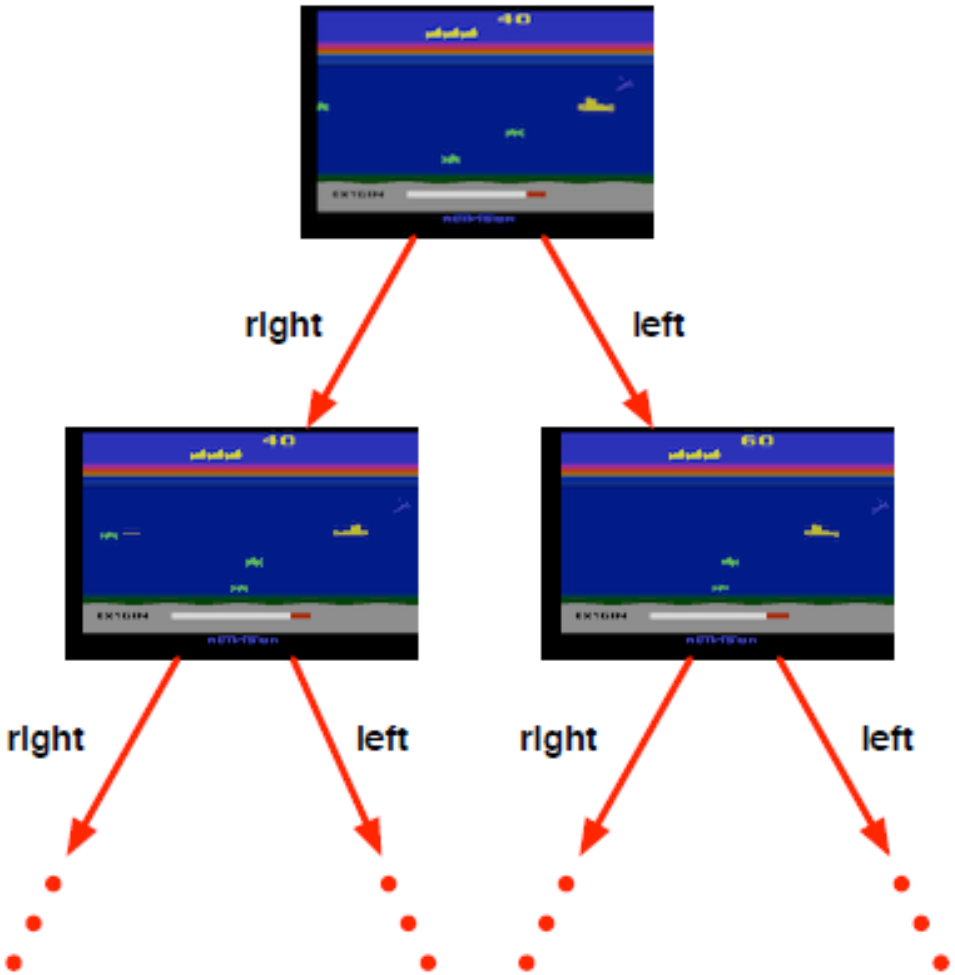


# Two Fundamental Problems in Sequential Decision Making

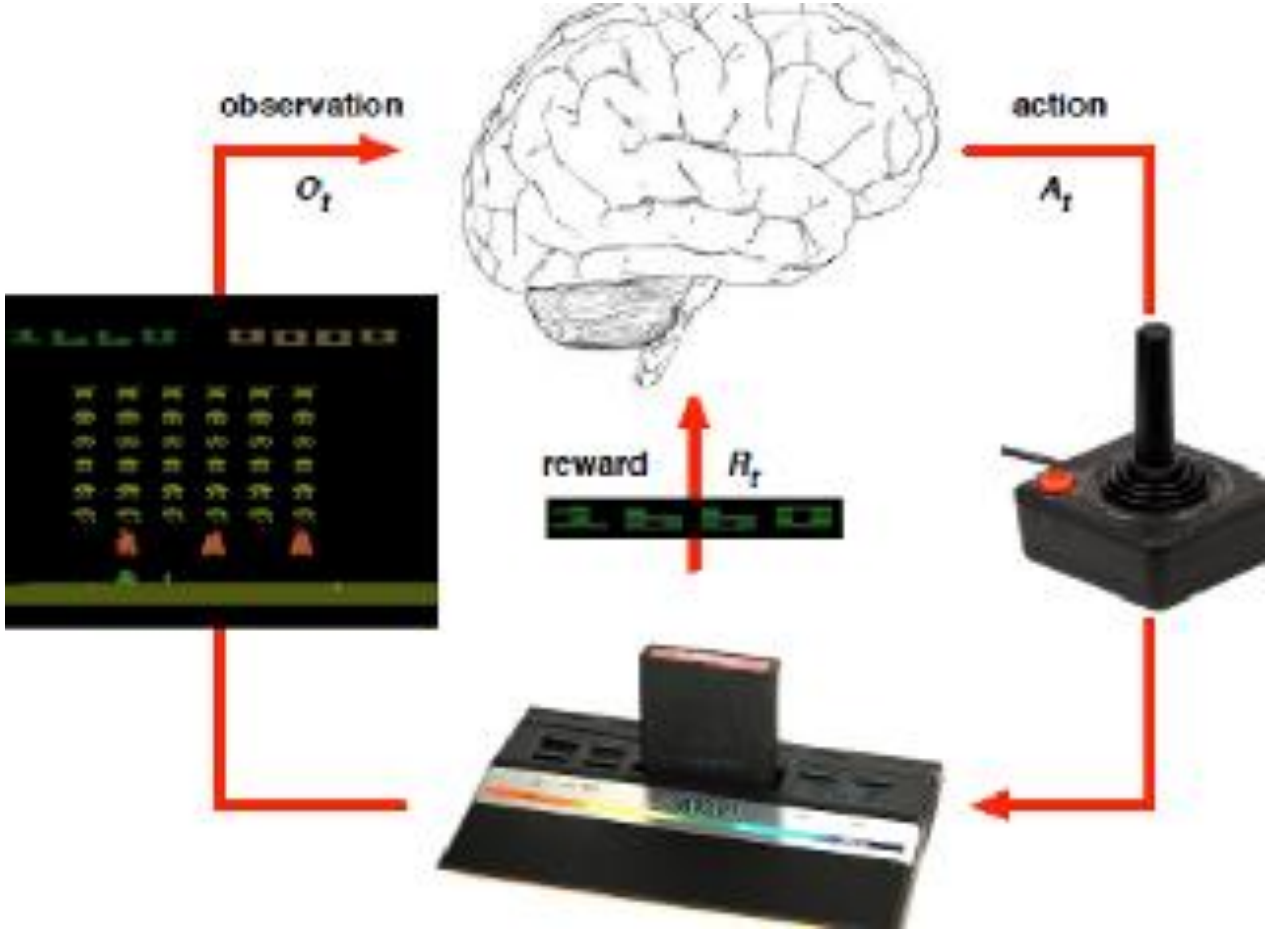
- **Planning**
  - Given model about how the environment works.
  - The agent performs computations with its model
  - The agent improves its policy
- **Reinforcement learning**
  - Agent doesn’t know how environment works
  - The agent interacts with the environment learn how environment works
  - The agent improves its policy (also involves planning)

# 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



# Atari Example: Reinforcement Learning




- Rules of the game are unknown
- Learn directly by taking actions and seeing what happens
- Try to find a good policy over time that yields high reward
- Planning is needed in inference or forward pass

# Exploration and Exploitation

- Agent only experiences what happens for the actions it tries!
- How should an RL agent balance its actions?
  - Exploration: trying new things that might enable the agent to make better decisions in the future
  - Exploitation: choosing actions that are expected to yield good reward given the past experience
- Often there may be an exploration-exploitation trade-off
  - May have to sacrifice reward in order to explore & learn about potentially better policy

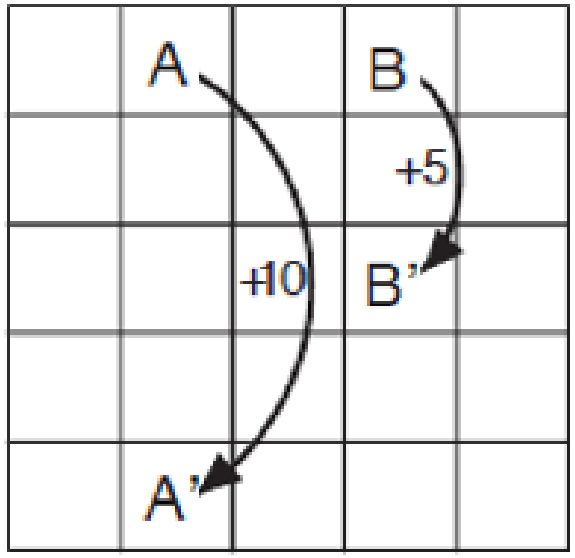
# Exploration and Exploitation

- Restaurant Selection
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Oil Drilling
  - Exploitation: Drill at the best-known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is
  - Exploration: play an experimental move

# Prediction and Control

- Prediction: evaluate the future
  - Given a policy
- Control: optimise the future
  - Find the best policy

# Gridworld Example: Prediction



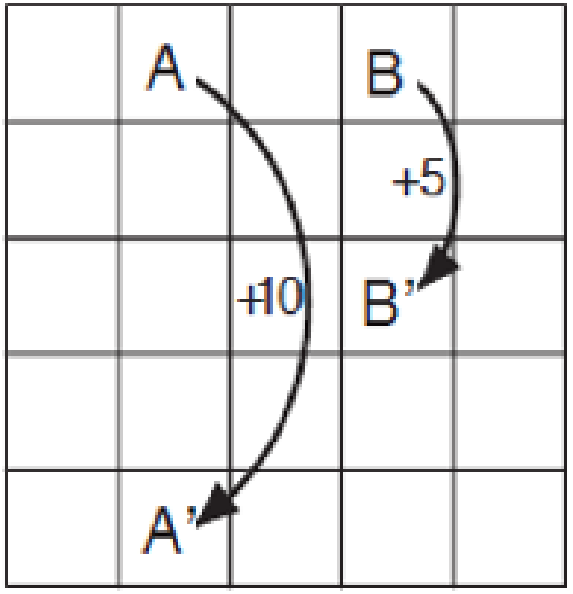

Gridworld


| | | | | |
|---|---|---|---|---|
| 3.3 | 8.8 | 4.4 | 5.3 | 1.5 |
| 1.5 | 3.0 | 2.3 | 1.9 | 0.5 |
| 0.1 | 0.7 | 0.7 | 0.4 | -0.4 |
| -1.0 | -0.4 | -0.4 | -0.6 | -1.2 |
| -1.9 | -1.3 | -1.2 | -1.4 | -2.0 |

$v$

What is the value function for the uniform random policy?

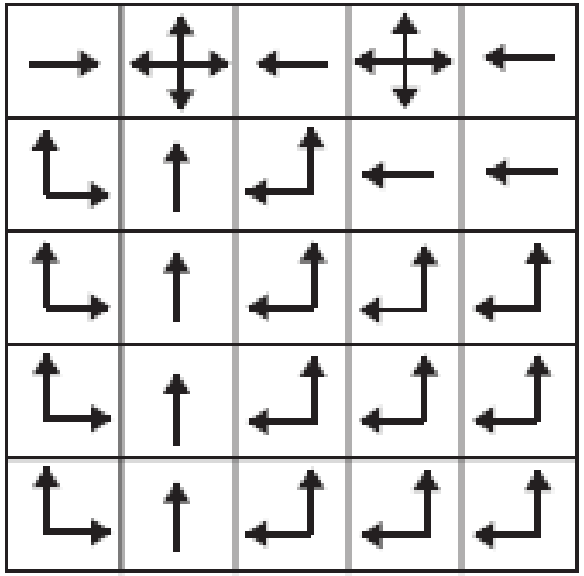
# Gridworld Example: Control




Gridworld

| | | | | |
|---|---|---|---|---|
| 22.0 | 24.4 | 22.0 | 19.4 | 17.5 |
| 19.8 | 22.0 | 19.8 | 17.8 | 16.0 |
| 17.8 | 19.8 | 17.8 | 16.0 | 14.4 |
| 16.0 | 17.8 | 16.0 | 14.4 | 13.0 |
| 14.4 | 16.0 | 14.4 | 13.0 | 11.7 |

$v_*$



$\pi_*$

What is the **optimal** value function over all possible policies?

What is the **optimal** policy?

# What can deep learning & RL do well now?

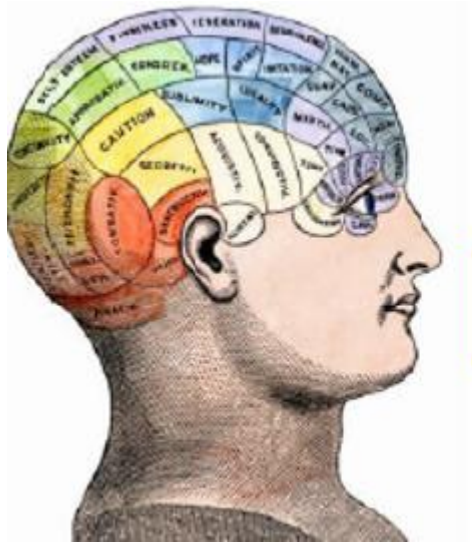
- Acquire high degree of proficiency in domains governed by simple, known rules
- Learn simple skills with raw sensory inputs, given enough experience
- Learn from imitating enough human provided expert behaviour

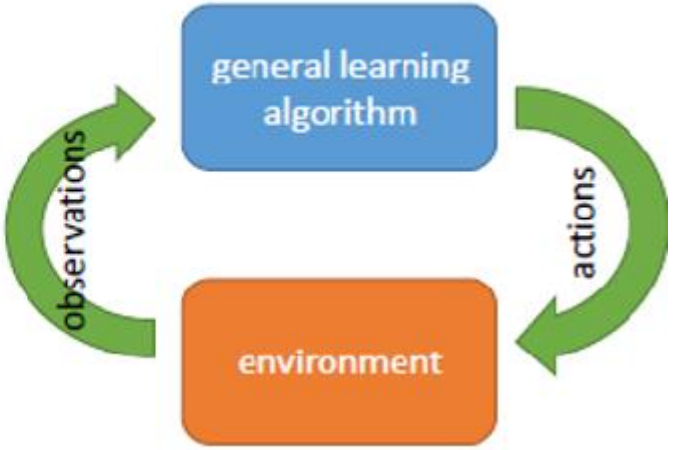

# What has proven challenging so far?

- Humans can learn incredibly quickly
  - Deep RL methods are usually slow
- Humans can reuse past knowledge
  - Transfer learning in deep RL is an open problem
- Not clear what the reward function should be
- Not clear what the role of prediction should be

Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.

- Alan Turing





# Experimenting with Reinforcement Learning


- Getting hand dirty on reinforcement learning is very important
- Deep learning and AI become more and more empirical
- Trial and error approach to learn reinforcement learning

# Coding

- Python
- Deep learning libraries: PyTorch or TensorFlow

# In case you need it

- Python tutorial: http://cs231n.github.io/python-numpy-tutorial/
- PyTorch tutorial: https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html