高等机器学习

强化学习

清华大学电子系
微软亚洲研究院

Microsoft

清華大學
Tsinghua University

Outline

- General introduction
- Basic settings
- Tabular approach
- Deep reinforcement learning
- Recent advances
- Challenges and opportunities
- Appendix: selected applications

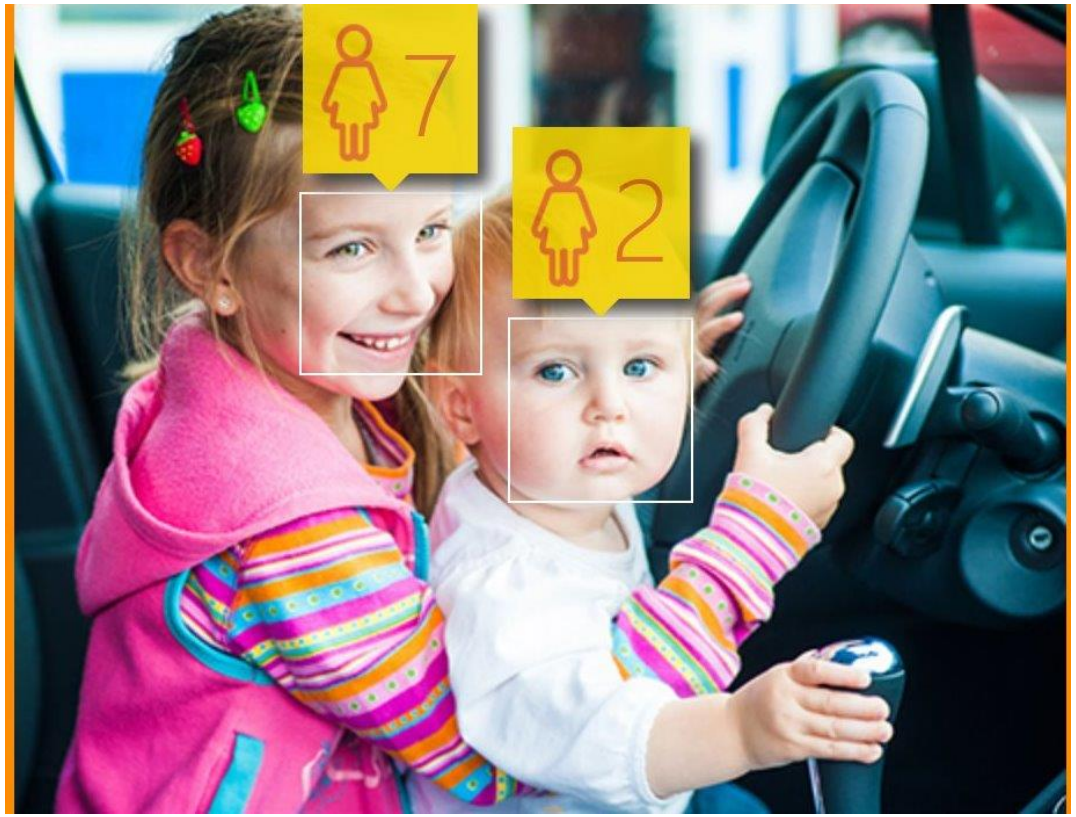
General Introduction

Machine Learning

Machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Supervised Learning

- Learn from labeled data
- Classification, regression, ranking



国内版 国际版

tsinghua ee

All Images Videos 翻译成中文 关闭取词

Microsoft See work results

471,000 Results Any time

Department of Electronic Engineering,Tsinghua University
www.tsinghua.edu.cn/publish/eeen
Tsinghua-Johns Hopkins University Dual Degree Mast... Associate Professor Peng Liangrui's research g... Faculty Openings - Electronic Engineering Dept. Ts...
Institute of Information System · Institute of Circuits & Systems · News · Institute of Communication

Department of Electrical Engineering,Tsinghua
www.tsinghua.edu.cn/publish/eeaen
电机系. Department of Electrical Engineering. search; Home; About; Admissions; Research; Teaching; service; staff
News · About · Admissions · Staff · Contact Us

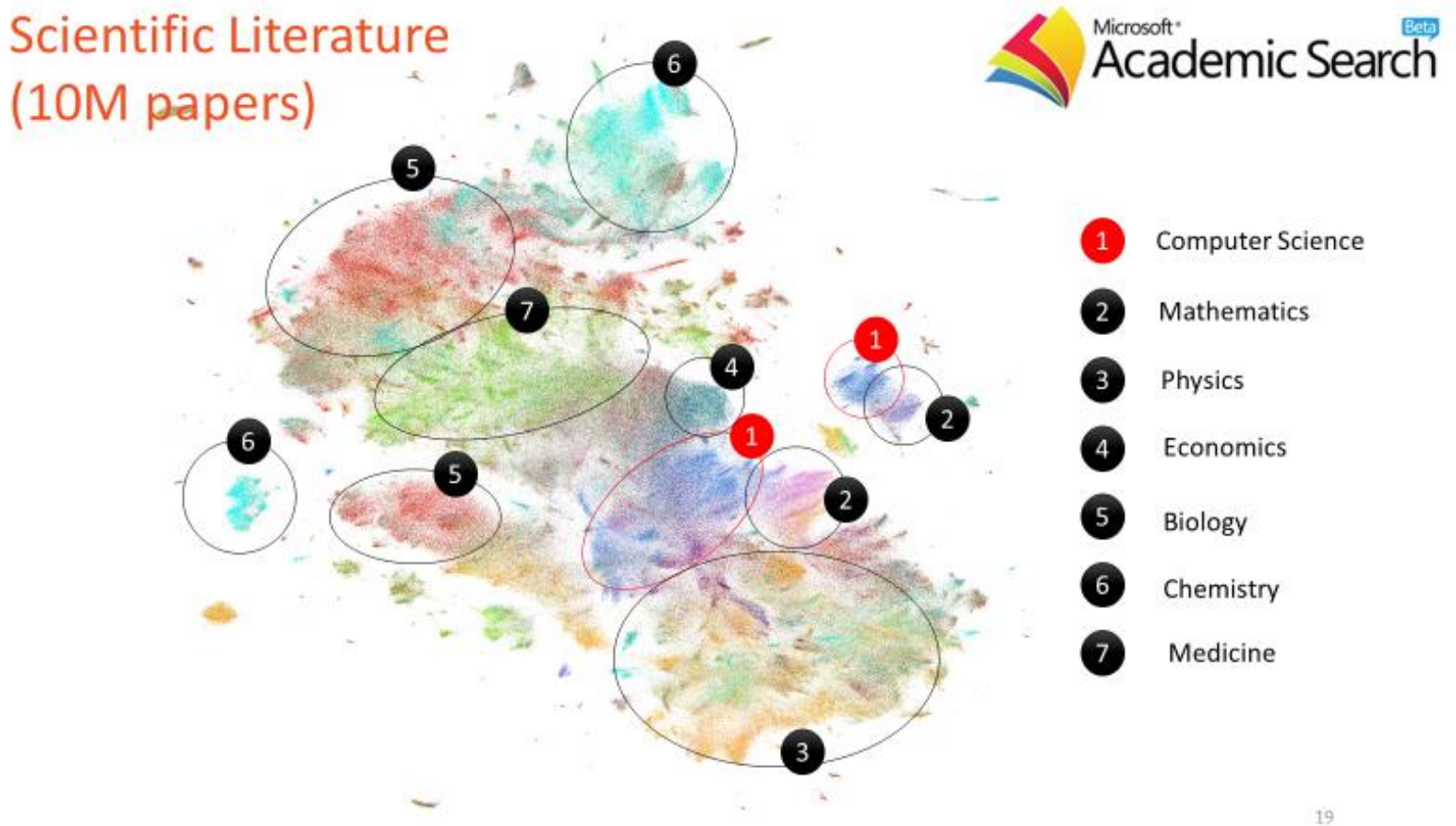
ee.tsinghua.edu.cn - 清华大学电子工程系
www.ee.tsinghua.edu.cn Translate this page
thu-jhu双硕士学位项目. 电子系举办第三十二期离退休教工沙龙活动. 电子系进行消防培训及疏散演练. 高校电子信息核心课程师资培训班（第一期）回顾

Department of Electrical Engineering,Tsinghua
https://www.tsinghua.edu.cn/publish/eeaen/1188/index.html
The Department of Electrical Engineering at Tsinghua University was founded in 1932. After nearly 80 years of development, it has established an excellent education and research reputation in China and around the world. More than 10,000 students have graduated.

Department of Electronic Engineering,Tsinghua University
www.ee.tsinghua.edu.cn/publish/eeen/3756/index.html
电子工程系. Department of Electronic Engineering . search; Home; About EE; Academics; Research; News; People; Contact

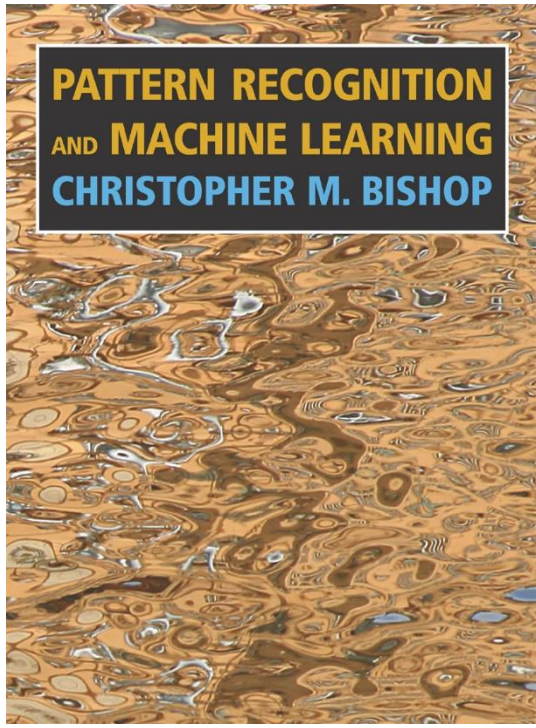
Unsupervised Learning

- Learn from unlabeled data, find structure from the data
- Clustering
- Dimension reduction



Reinforcement Learning

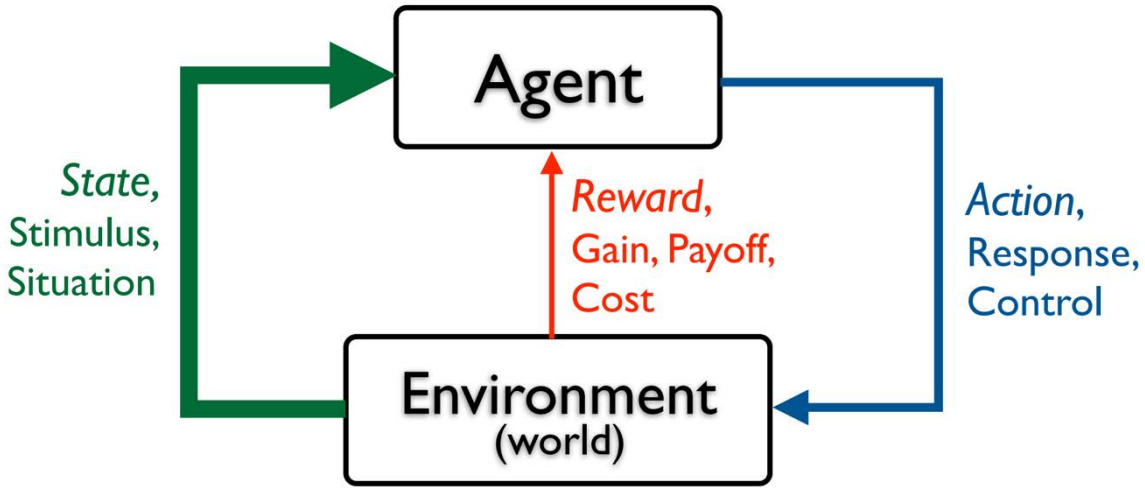
The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning….

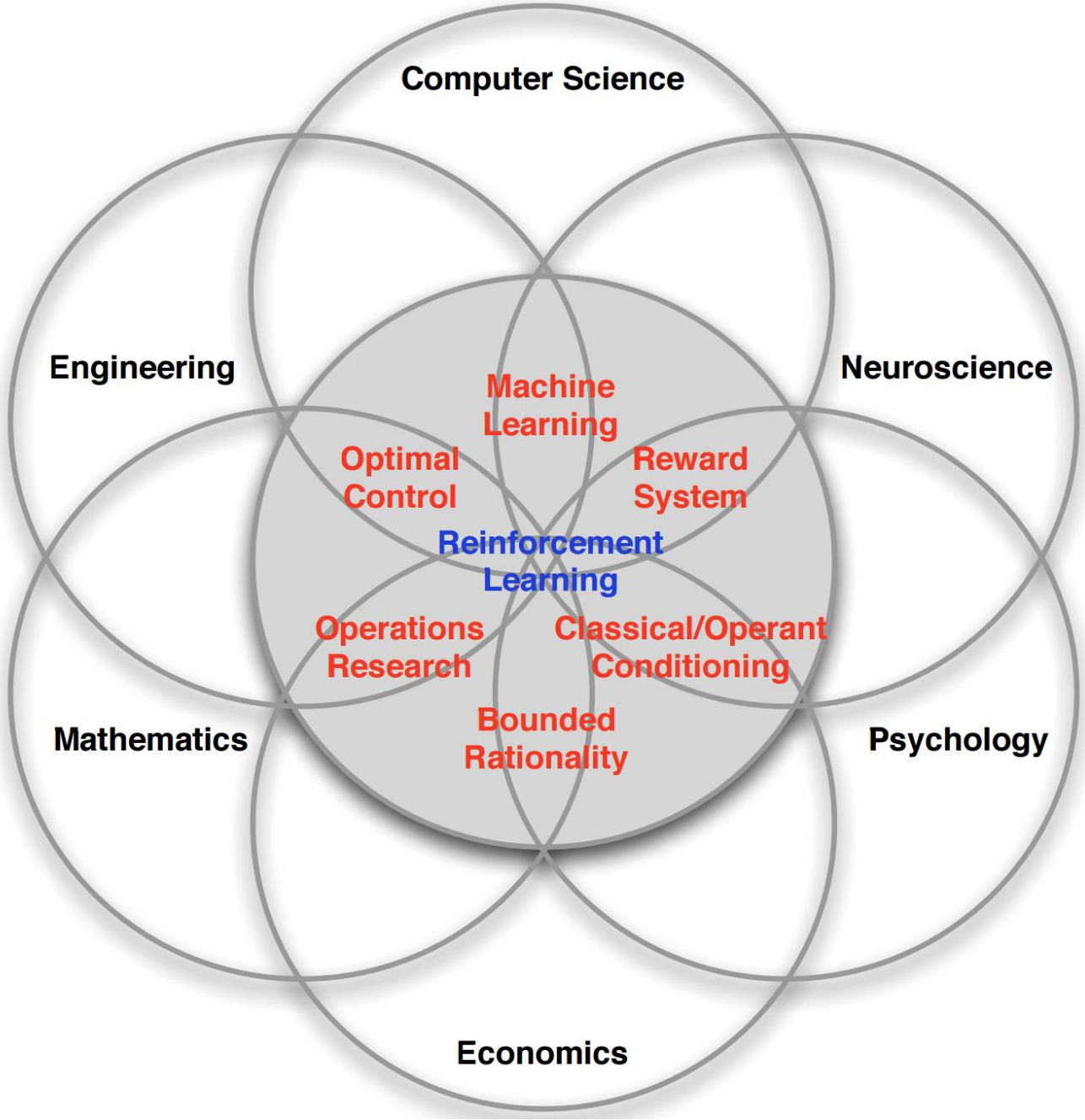


Reinforcement learning problems involve learning what to do - how to map situations to actions - so as to maximize a numerical reward signal.

Reinforcement Learning

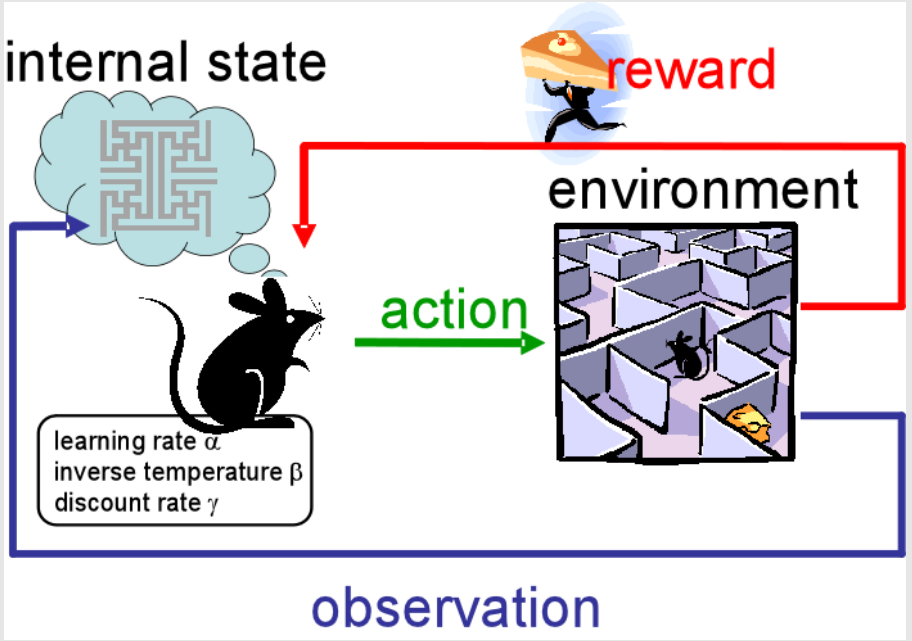
- Agent-oriented learning-learning by interacting with an environment to achieve a goal
  - Learning by trial and error, with only delayed evaluative feedback(reward)
  - Agent learns a policy mapping states to actions
    - Seeking to maximize its cumulative reward in the long run




Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics

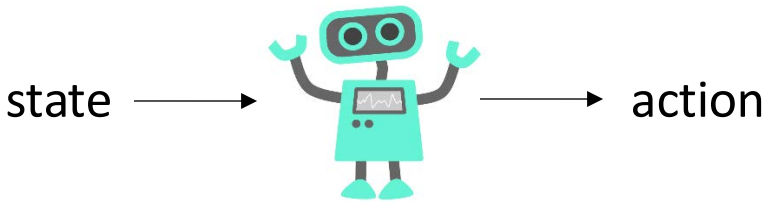
RL vs Other Machine Learning

- Supervised learning
  - Regression, classification, ranking, …
  - Learning from examples, learning from a teacher
- Unsupervised learning
  - Dimension reduction, density estimation, clustering
  - Learning without supervision
- Reinforcement learning
  - Learning from interaction, learning by doing, learning from delayed reward
  - Sequential decision making

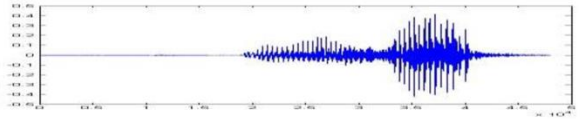


One-shot Decision v.s Sequential Decisions

- Agent Learns a Policy

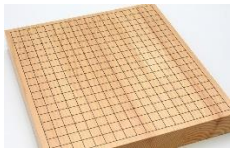


- Supervised Learning

$f($  $) =$ “How are you”

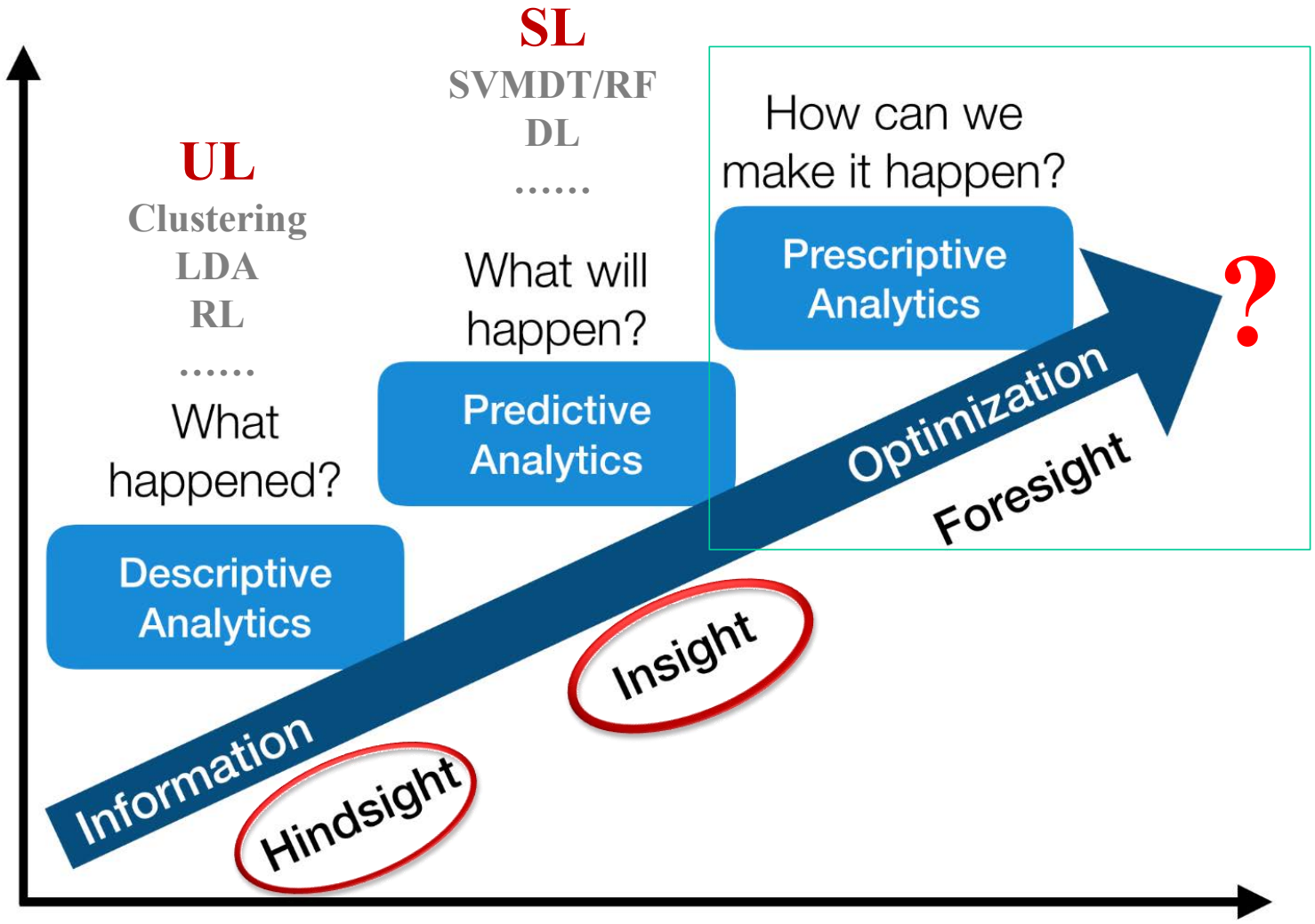
$f($  $) =$ “Cat”

- Reinforcement Learning

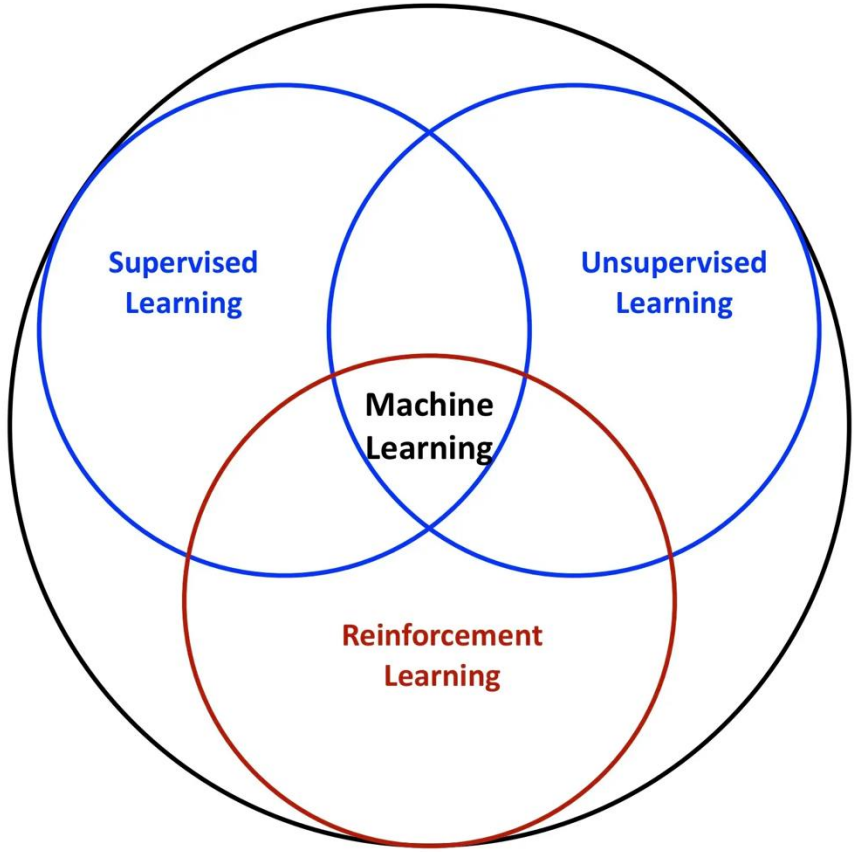




When to Use Reinforcement Learning

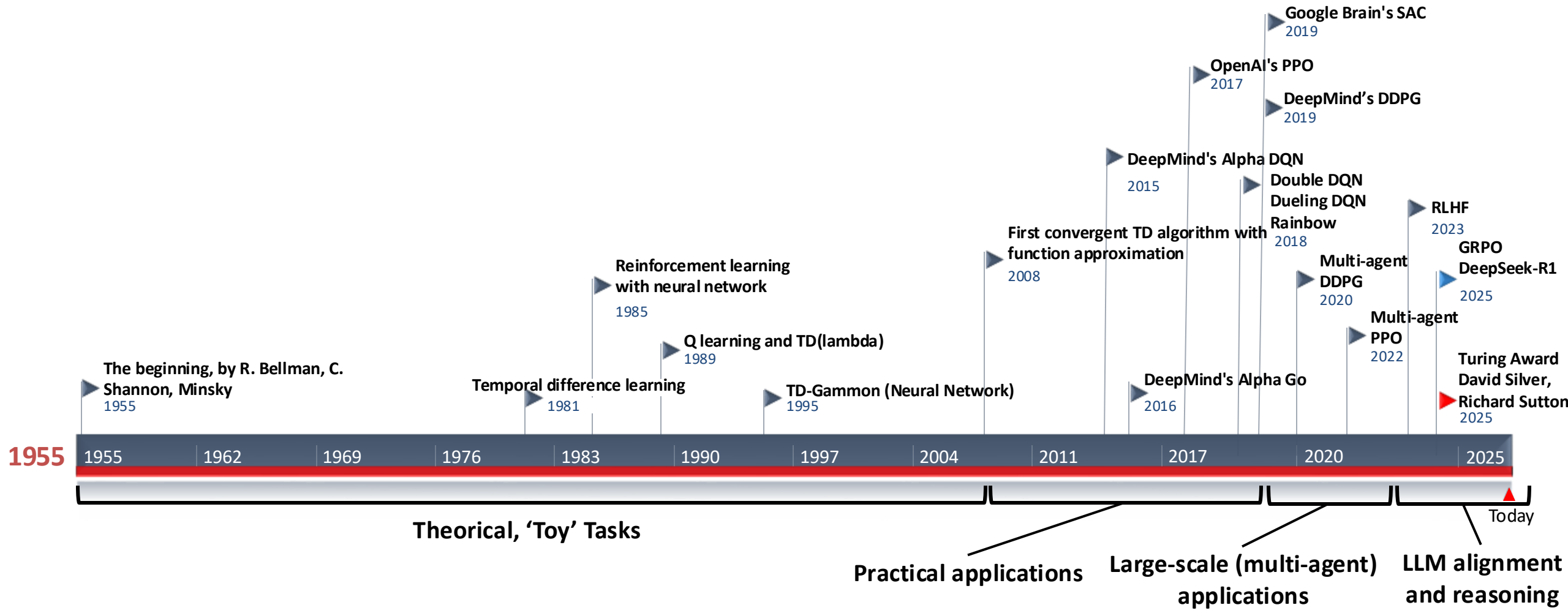


http://www.argmin.net/2018/01/29/taxonomy/



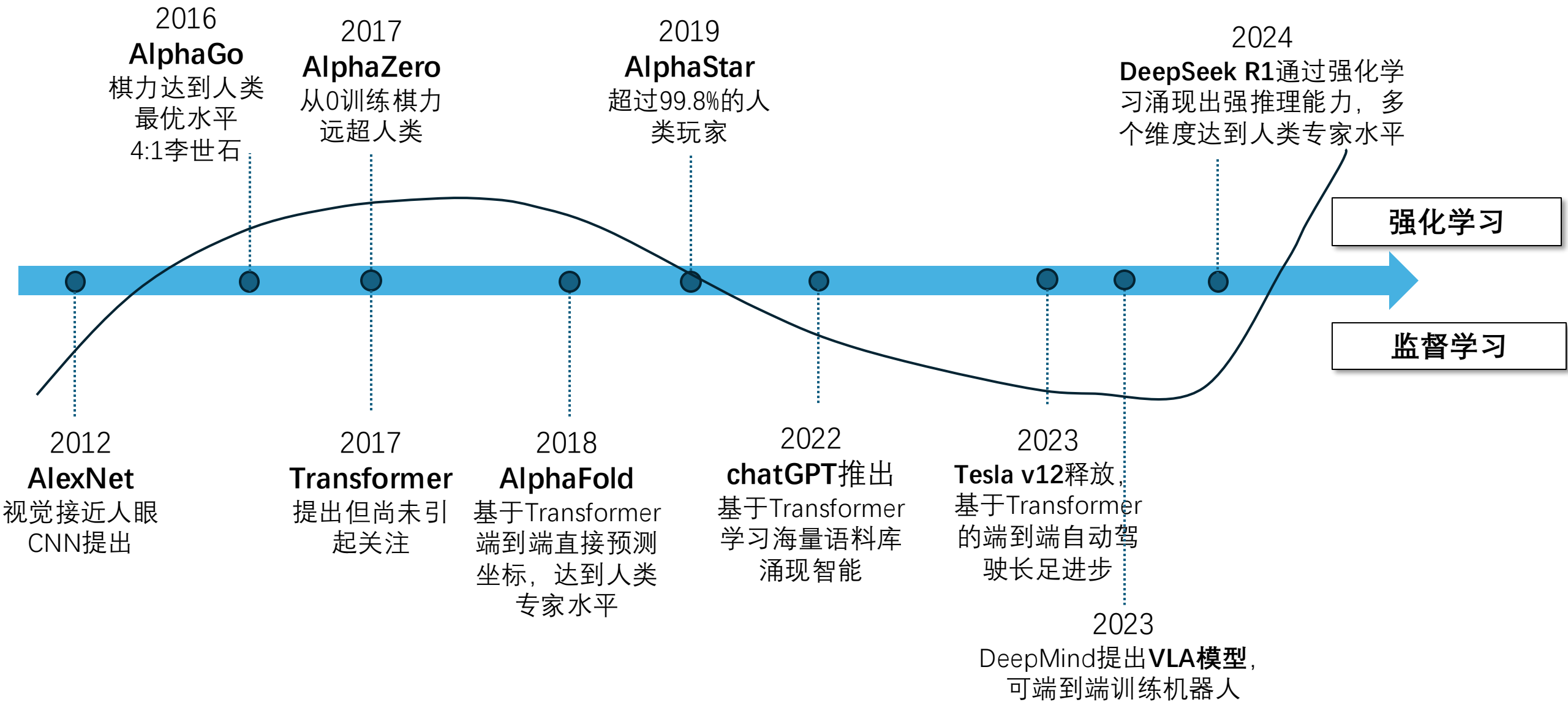
When to Use Reinforcement Learning

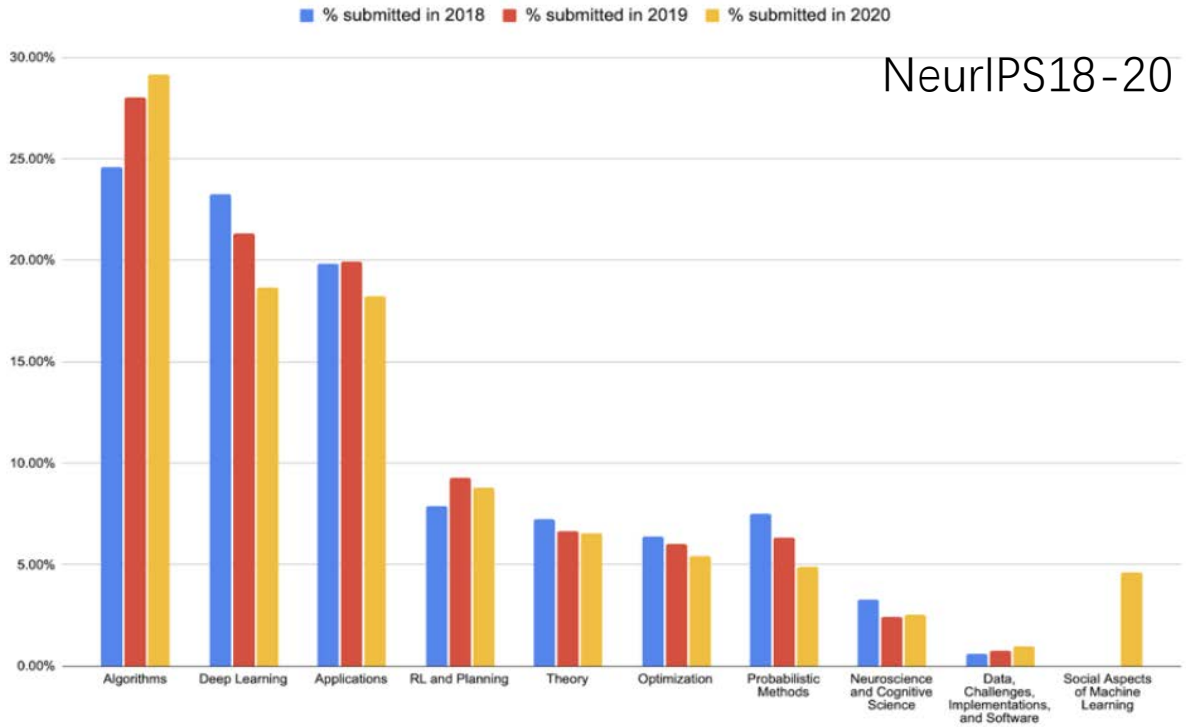
- Second order effect : your output (action) will influence the data (environment)
  - Web click : You learn from your observed CTRs, if you adapt a new ranker, the observed data distribution will change.
  - City traffic : You give a current best strategy to the traffic jam, but it may cause larger jam in other place that you don’t expect
  - Financial market
- Tasks : You focus on long-term reward from interactions, feedback
  - Job market
- Psychology learning : understanding user’s sequential behavior
  - Social Network : why does he follow this guy, for linking new friends, for own interests


Google Brain's SAC
2019
OpenAI's PPO
2017
DeepMind's DDPG
2019
DeepMind's Alpha DQN
2015
Double DQN
Dueling DQN
Rainbow
2018
RLHF
2023
First convergent TD algorithm with function approximation
2008
Reinforcement learning with neural network
1985
Multi-agent DDPG
2020
GRPO
DeepSeek-R1
2025
Q learning and TD(lambda)
1989
Multi-agent PPO
2022
The beginning, by R. Bellman, C. Shannon, Minsky
1955
Temporal difference learning
1981
TD-Gammon (Neural Network)
1995
DeepMind's Alpha Go
2016
Turing Award
David Silver, Richard Sutton
2025
1955
1955
1962
1969
1976
1983
1990
1997
2004
2011
2017
2020
2025
Today
Theorical, 'Toy' Tasks
Practical applications
Large-scale (multi-agent) applications
LLM alignment and reasoning

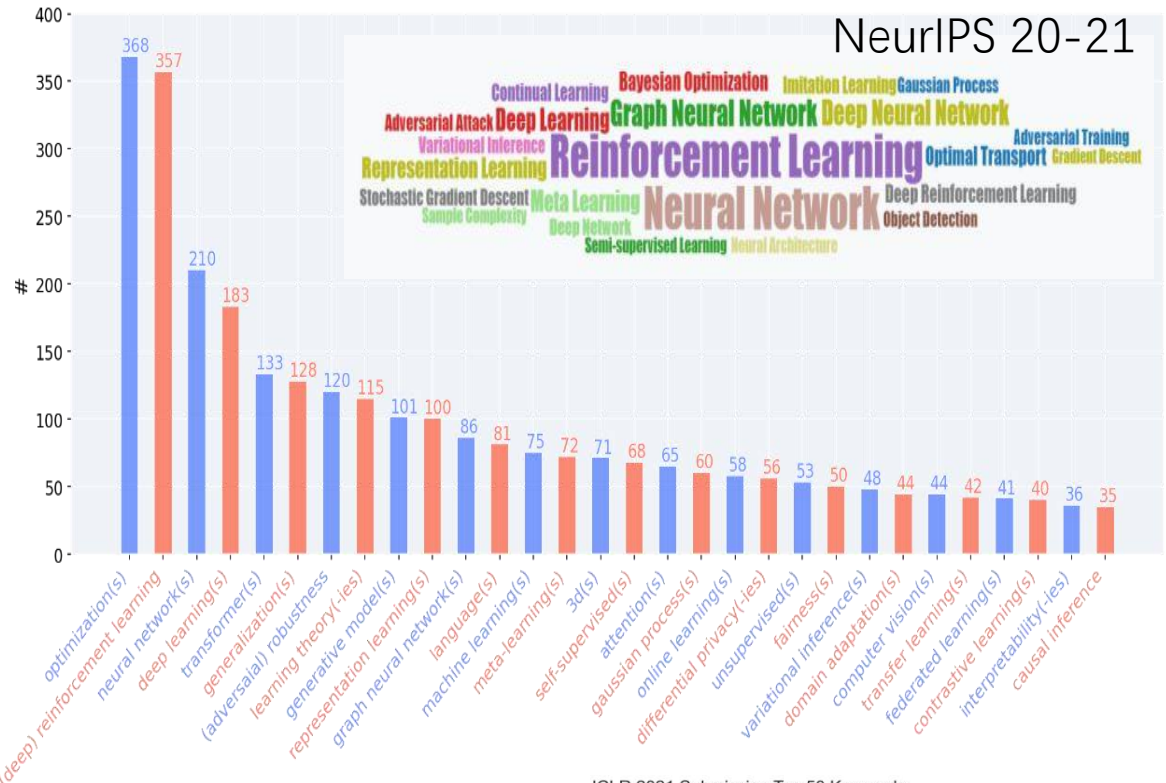
RL has achieved a wide of success across different applications.

The Important Role of RL: For the Past 10 Years of AI



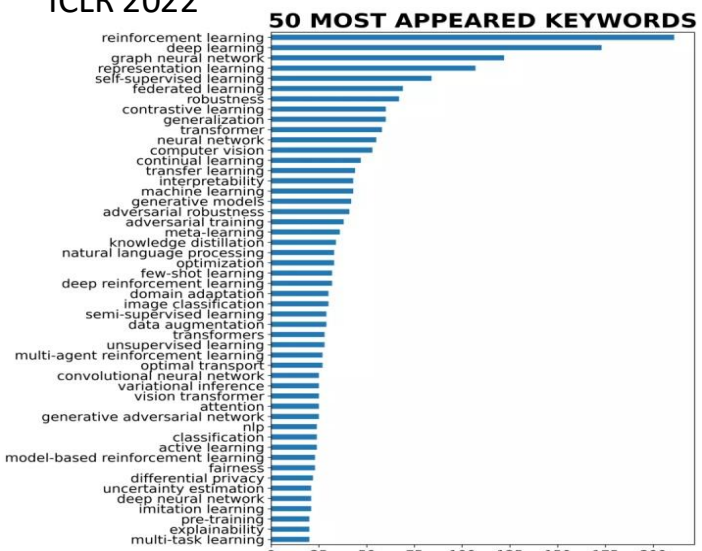


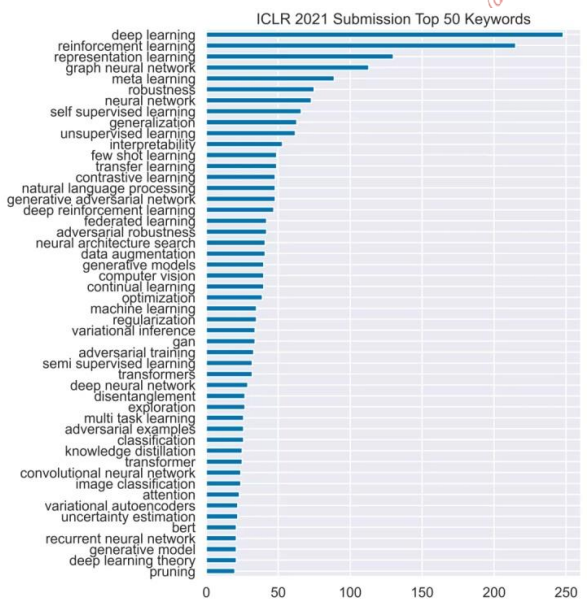
NeurIPS18-20

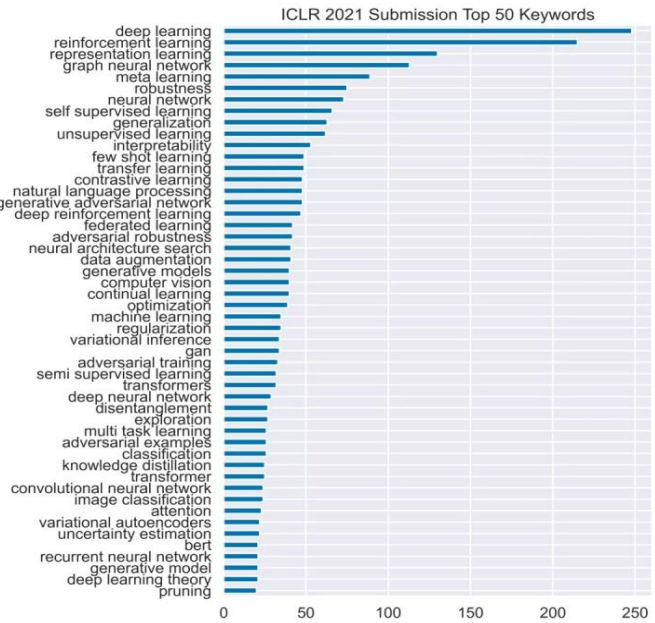


NeurIPS 20-21

ICLR 2022



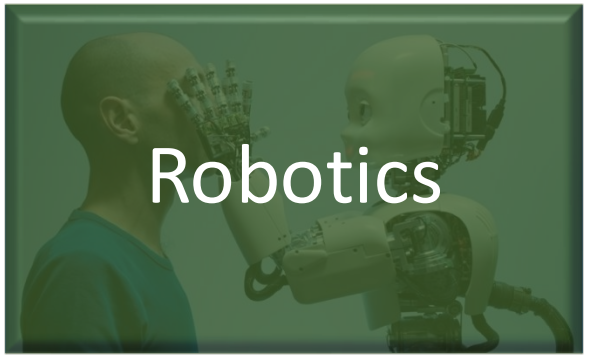




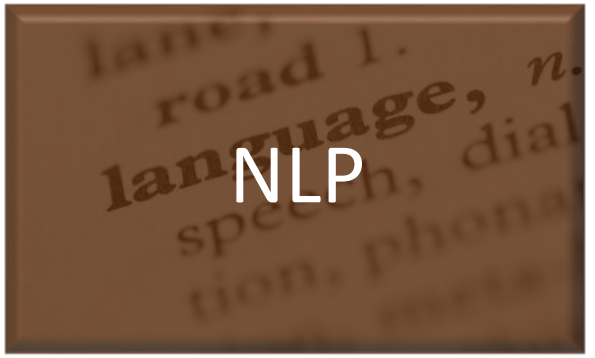

AlphaGo
Game


Neuro Science


Music & Movie


Robotics


Trading


NLP

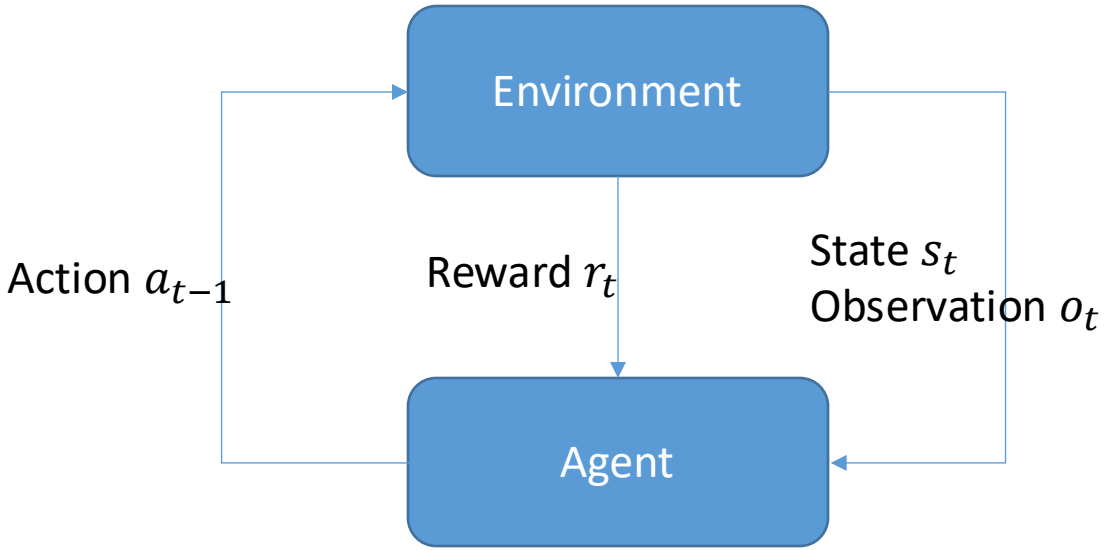

Healthcare


Education

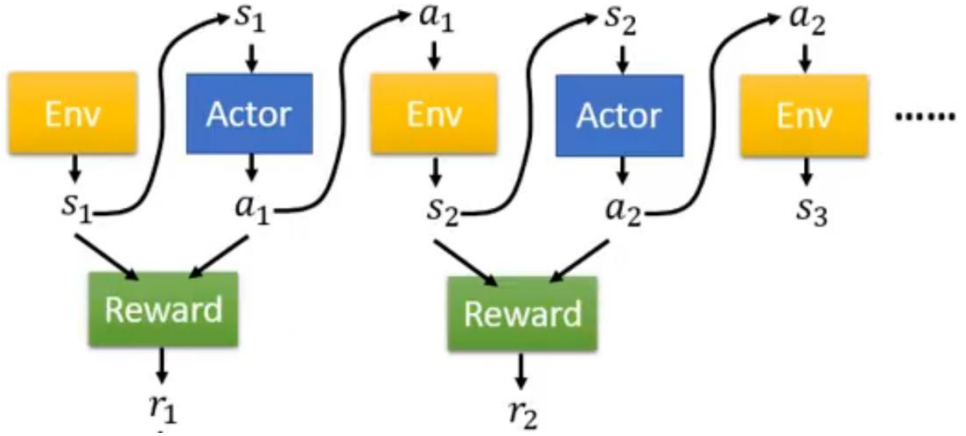

Control

Basic Settings

Reinforcement Learning: Actor, Envi., Reward



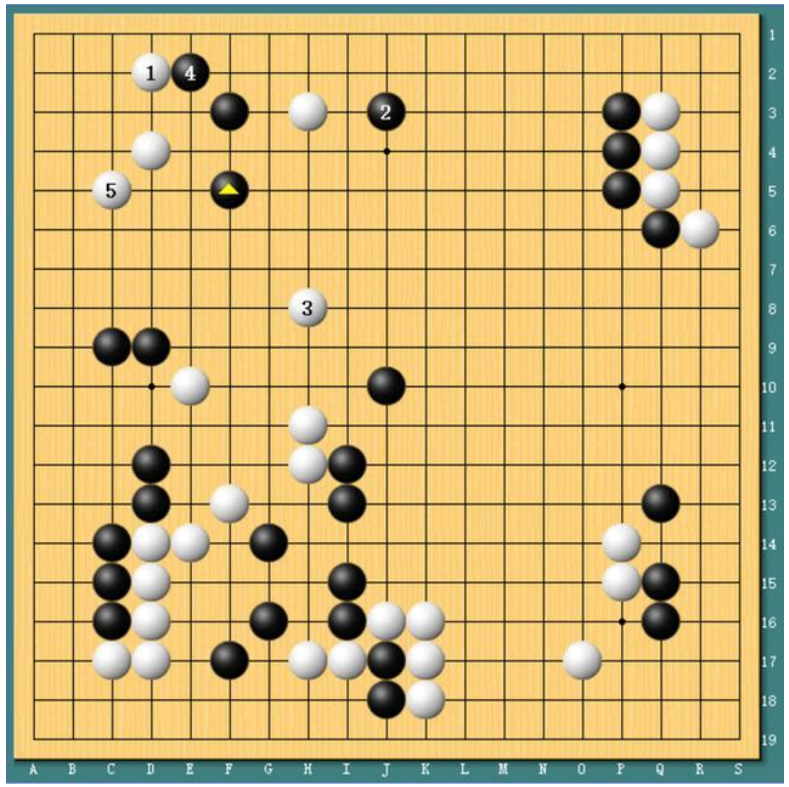
- a set of environment states S;
- a set of actions A;
- rules of transitioning between states;
- rules that determine the scalar immediate reward of a transition; and



Goal: Maximize expected long-term reward, always called return

Example Applications

Application	Action	Observation	State	Reward
Playing Go (boardgame)	Where to place a stone	Configuration of board	Configuration of board	Win game: +1 Else: -1



Example Applications

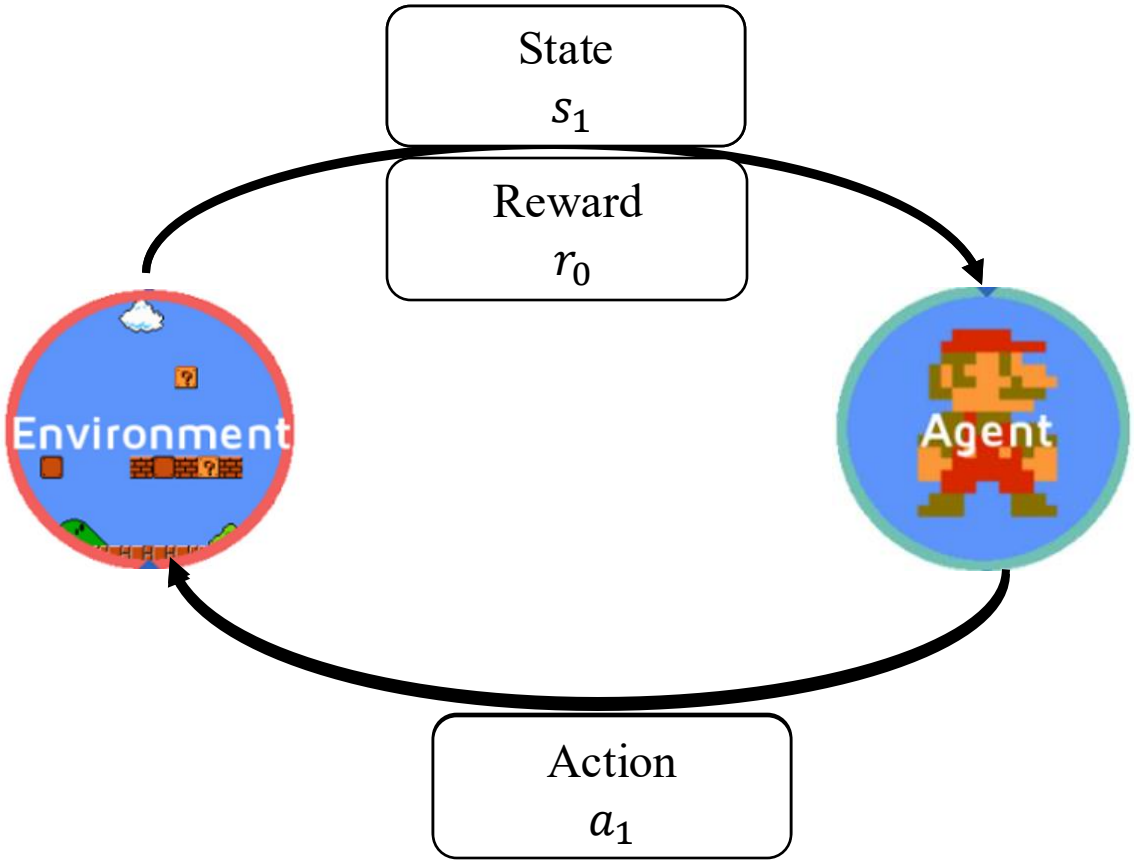
Application	Action	Observation	State	Reward
Playing Go (boardgame)	Where to place a stone	Configuration of board	Configuration of board	Win game: +1 Else: -1
Playing Atari (video games)	Joystick and button inputs	Screen at time *t*	Screen at times *t, t-1, t-2, t-3*	Game score increment

Example Applications

Application	Action	Observation	State	Reward
Playing Go (boardgame)	Where to place a stone	Configuration of board	Configuration of board	Win game: +1 Else: -1
Playing Atari (video games)	Joystick and button inputs	Screen at time *t*	Screen at times *t, t-1, t-2, t-3*	Game score increment
Conversational system	What to say to the user	What user says	History of the conversation	Task success: +10 Task fail: -20 Else: -1

Markov Decision Process

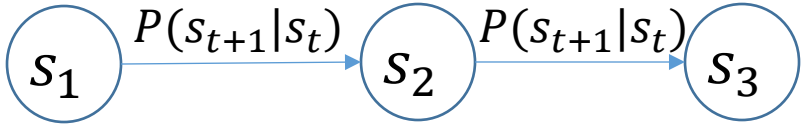
$$
\begin{aligned}
s_o &\sim p_0(s), \\
a_0 &\sim \pi(a|s_0), \\
r_0 &= R(s_0, a_0), \\
s_1 &\sim P(s|s_0, a_0), \\
a_1 &\sim \pi(a|s_1) \\
&\cdots, \\
s_t, \\
a_t, \\
r_t
\end{aligned}
$$



Markov Chain

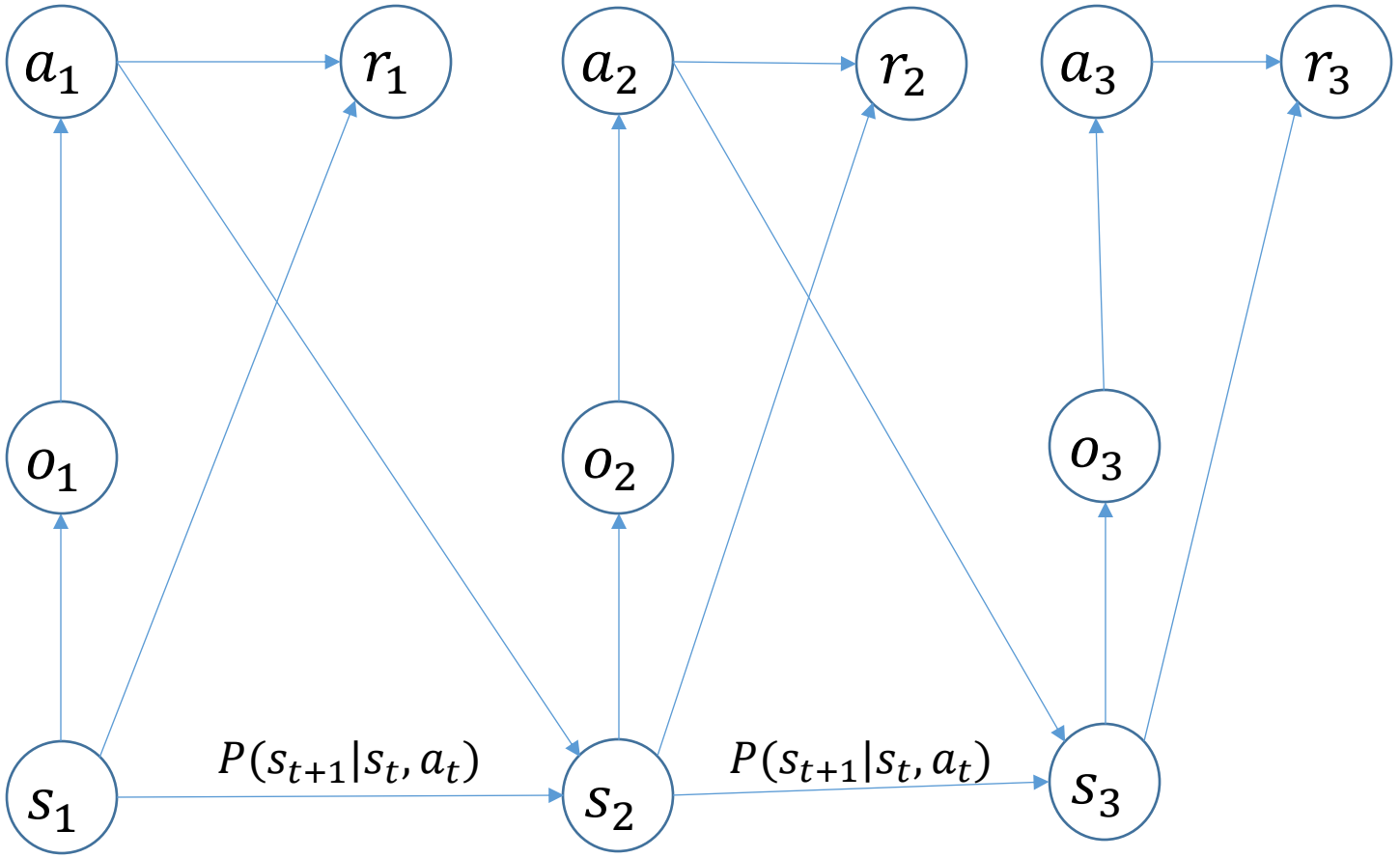
- Markov state

$$P(s_{t+1}|s_1, \dots, s_t) = P(s_{t+1}|s_t)$$



Andrey Markov

Markov Decision Process

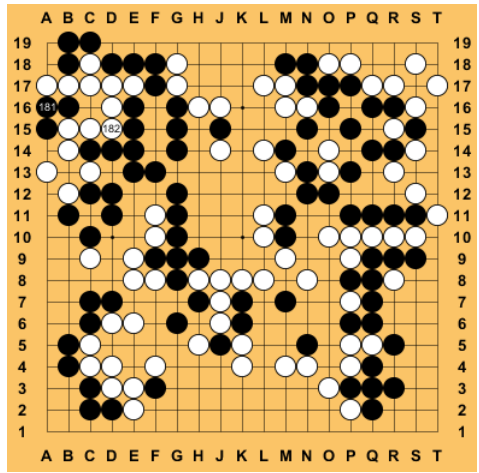


- s_t: state
- o_t: observation
- a_t: action
- r_t: reward

Markov Decision Process

- Fully observable environments ➔ Markov decision process (MDP)

$$o_t = s_t$$



- Partially observable environments ➔ partially observable Markov decision process (POMDP)

$$o_t \neq s_t$$

Markov Decision Process

- A Markov Decision Process(MDP) is a tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$
  - $\mathcal{S}$ is a finite set of states
  - $\mathcal{A}$ is a finite set of actions
  - $\mathcal{P}$ is state transition probability

$$p(s'|s,a) = \mathbf{Pr}\{S_{t+1} = s' \mid S_t = s, A_t = a\}$$

  - $\mathcal{R}$ is reward function

$$r(s,a,s') = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s']$$

  - γ is a discount factor $\gamma \in [0,1]$
- Trajectory.
  - $\ldots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \ldots$

Policy

- A mapping from state to action
  - Deterministic $a = \pi(s)$
  - Stochastic $a \sim \pi(a|s)$

- Informally, we are searching a policy to maximize the discounted sum of future rewards:

$$\text{to choose each } A_t \text{ to maximize } R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

Action-Value Function

- An action-value function says how good it is to be in a state, take an action, and thereafter follow a policy:

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \,\middle|\, S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\right]$$

Delayed reward is taken into consideration.

Action-Value Function

- An action-value function says how good it is to be in a state, take an action, and thereafter follow a policy:

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \,\middle|\, S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\right]$$

Delayed reward is taken into consideration.

- Action-value functions decompose into Bellman expectation equation.

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \,\middle|\, S_t = s, A_t = a, A_{t+1} \sim \pi\right]$$

Optimal Value Functions

- An optimal value function is the maximum achievable value.

$$q_{\pi_*}(s,a) = \max_{\pi} q_\pi(s,a) = q_*(s,a)$$

- Once we have q_* we can act optimally,

$$\pi_*(s) = \arg\max_{a} q_*(s,a)$$

- Optimal values decompose into Bellman optimality equation.

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right]$$

Review: Major Concepts of a RL Agent

- Model: characterizes the environment/system
  - State transition rule: $P(s'|s,a)$
  - Immediate reward: $r(s,a)$
- Policy: describes agent's behavior
  - a mapping from state to action, $\pi: S \Rightarrow A$
  - Could be deterministic or stochastic
- Value: evaluates how good is a state and/or action
  - Expected discounted long-term payoff
  - $v_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s]$
  - $q_\pi(s,a) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s, a_t = a]$

Tabular Approaches

Fundamental problems

- From settings
  - Planning: A model of the environment is known
  - Reinforcement learning: The environment is initially unknown
- From tasks
  - Evaluation: Given a policy, evaluate its quality
  - Control: Find the best policy

Before I tell you

- What things should we do before we can get the best policy?
- What indicator can I use to represent the quality of a policy?
- How can we calculate it?
- Can we calculate it smarter?
- And then?

Learning and Planning

- Two fundamental problems in sequential decision making
- 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
- Reinforcement learning:
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy, with exploring the environment

Recall: Bellman Expectation Equation

- State-value function

$$v_\pi(s) = E_\pi\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s\}$$
$$= E_\pi[r_{t+1} + \gamma v_\pi(s') | s]$$
$$= r\big(s, \pi(s)\big) + \gamma \sum_{s'} P(s'|s, \pi(s)) v_\pi(s')$$

$$v_\pi = r_\pi + \gamma P_\pi v_\pi$$

- Action-value function

$$q_\pi(s, a) = E_\pi[r_{t+1} + \gamma q_\pi(s', a') | s, a]$$

Richard Bellman

Recall: Bellman Optimality Equation

- Optimal value function
  - Optimal state-value function: $v_*(s) = \max_\pi v_\pi(s)$
  - Optimal action-value function: $q_*(s,a) = \max_\pi q_\pi(s,a)$
- Bellman optimality equation
  - $v_*(s) = \max_a q_*(s,a)$
  - $q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s')$

Planning (Policy Evaluation)

Given an exact model (i.e., reward function, transition probabilities,), and a fixed policy π

Algorithm:

Arbitrary initialization: v_0

For $k = 0,1,2, \ldots$

$$v_\pi^{k+1} = r_\pi + \gamma P_\pi v_\pi^k$$

Stopping criterion: $\left|v_\pi^{k+1} - v_\pi^k\right| \leq \epsilon$

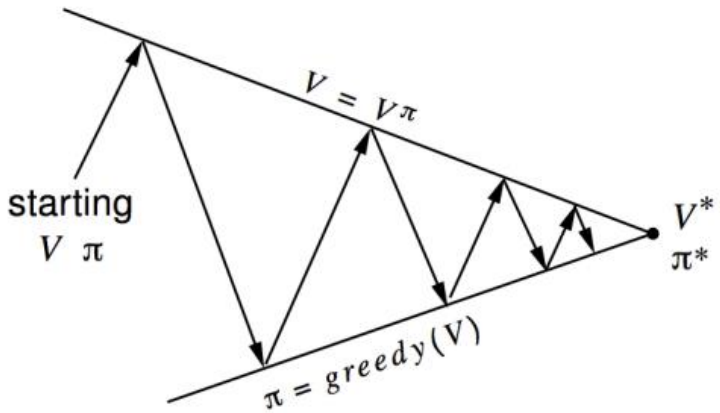
Planning (Optimal Control)

Given an exact model (i.e., reward function, transition probabilities)

Two ways:

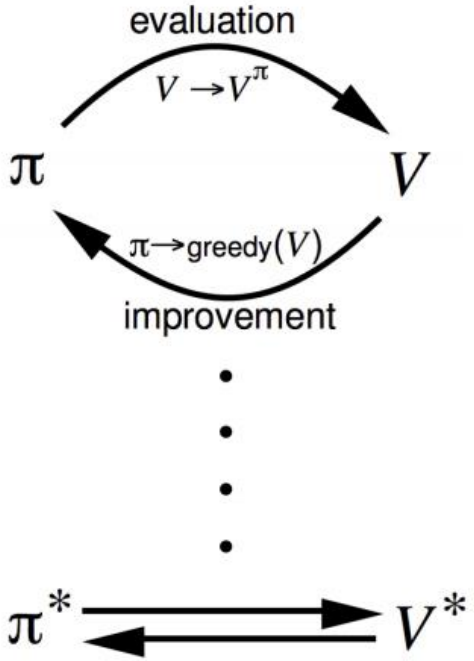
- Policy iteration
- Value iteration

Policy Iteration



Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement



$$v_*(s) = \max_a q_*(s,a)$$

Value iteration

- Synchronous value iteration stores two copies of value function

  for all s in $\mathcal{S}$

$$v_{new}(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{old}(s') \right)$$

  $v_{old} \leftarrow v_{new}$

- In-place value iteration only stores one copy of value function

  for all s in $\mathcal{S}$

$$v(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right)$$

Learning in MDPs

- Have access to the real system but no model
- Generate experience $o_1, a_1, r_1, o_2, a_2, r_2, \ldots, o_{t-1}, a_{t-1}, r_{t-1}, o_t$

- Two kinds of approaches
  - Model-free learning
  - Model-based learning

Monte-Carlo Policy Evaluation

Return : $G_t = \sum_{i=1}^{T-t} \gamma^i R_{t+i}$

Value function : $v_\pi(s) = E_\pi(G_t | S_t = s)$

Action value function : $q_\pi(s, a) = E(G_t | S_t = s, A_t = a)$

Incremental Monte-Carlo Update

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right)$$
$$= \frac{1}{k}(x_k + (k-1)\mu_{k-1})$$
$$= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

For each state s with return G_t: $N(s) \leftarrow N(s) + 1$

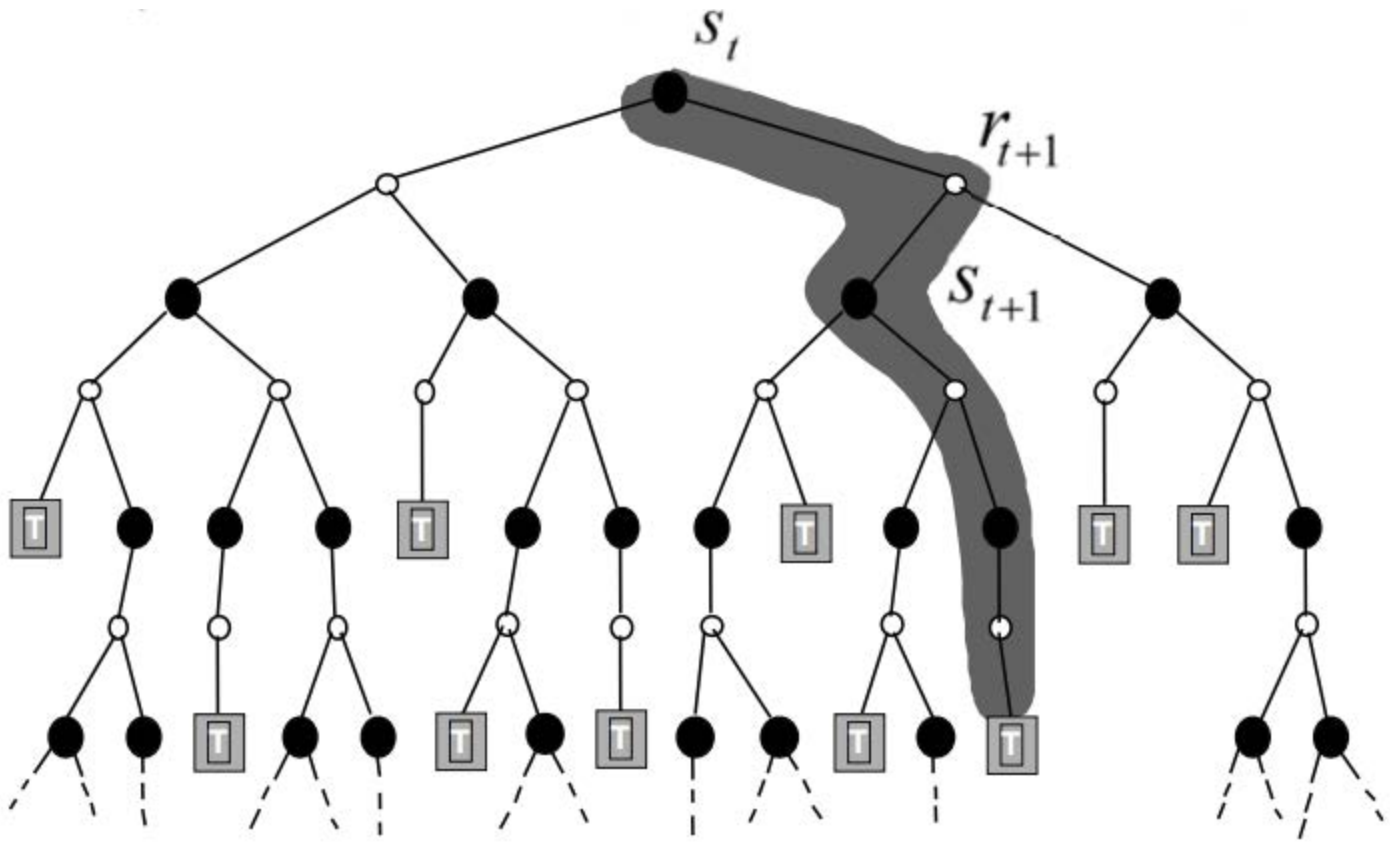
$$V(s) \leftarrow V(s) + \frac{1}{N(s)}(G_t - V(s))$$

Handle non-stationary problem: $V(s) \leftarrow V(s) + \alpha(G_t - V(s))$

Monte-Carlo Policy Evaluation

$$v(s_t) \leftarrow v(s_t) + \alpha[G_t - v(s_t)]$$

G_t is the actual long-term return following state s_t in a sampled trajectory



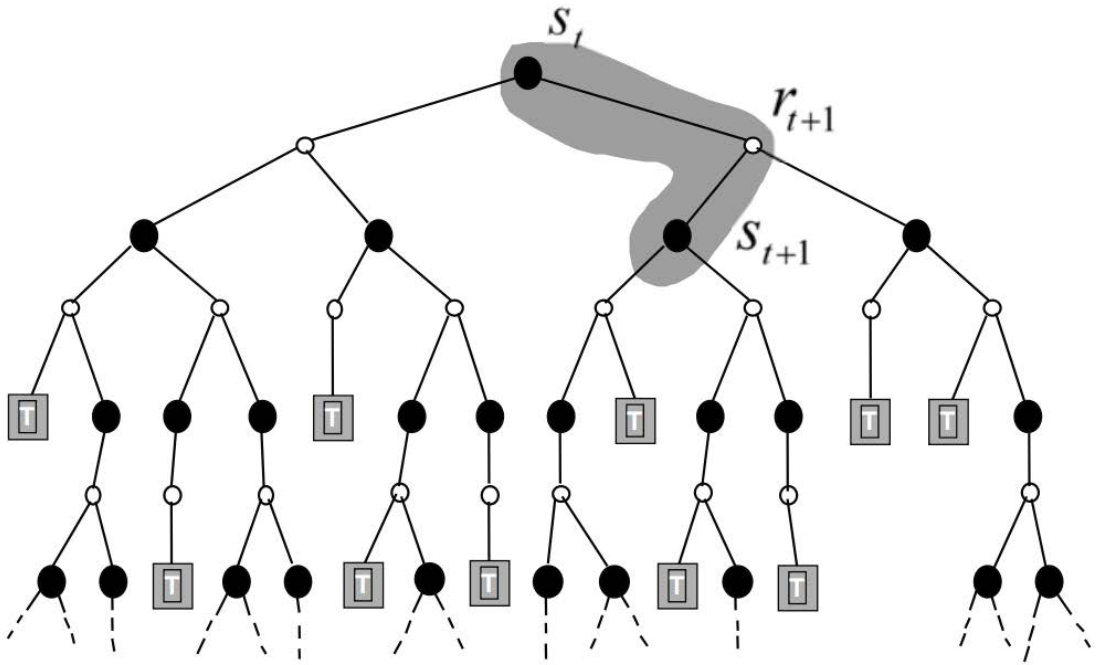
Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is model-free: no knowledge of MDP transitions / rewards
- MC learns from complete episodes
  - Values for each state or pair state-action are updated only based on final reward, not on estimations of neighbor states
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to episodic MDPs
  - All episodes must terminate

Temporal-Difference Policy Evaluation

Monte-Carlo : $v(s_t) \leftarrow v(s_t) + \alpha[G_t - v(s_t)]$

TD: $v(s_t) \leftarrow v(s_t) + \alpha[r_{t+1} + \gamma v(s_{t+1}) - v(s_t)]$
r_t is the actual immediate reward following state s_t in a sampled step



Temporal-Difference Policy Evaluation

- TD methods learn directly from episodes of experience
- TD is model-free: no knowledge of MDP transitions / rewards
- TD learns from incomplete episodes, by bootstrapping
- TD updates a guess towards a guess

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $v(s_t)$ toward estimated return $r_{t+1} + \gamma v(s_{t+1})$

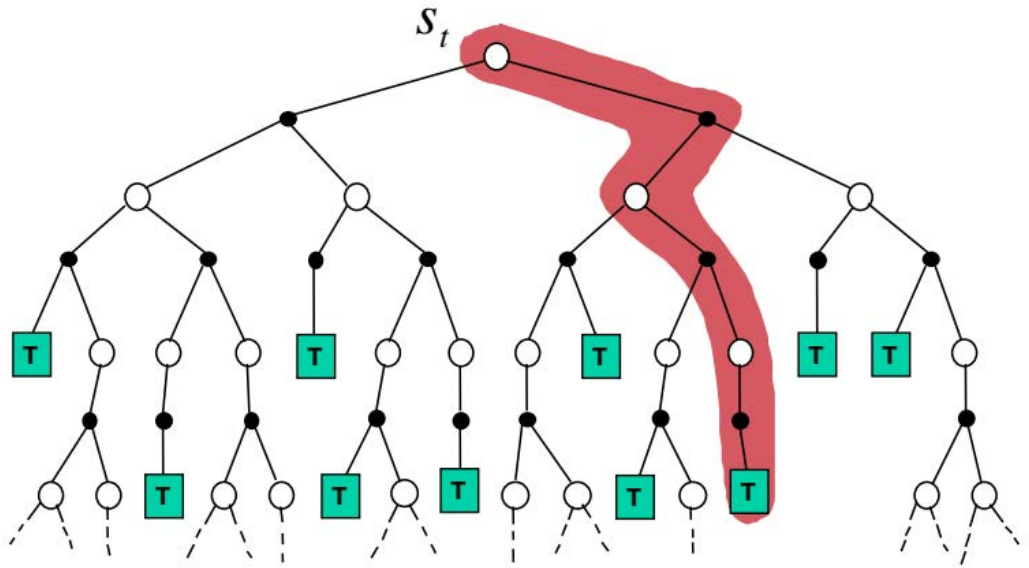
$$v(s_t) = v(s_t) + \alpha(r_{t+1} + \gamma v(s_{t+1}) - v(s_t))$$

  - $r_{t+1} + \gamma v(s_{t+1})$ is called the TD target
  - $\delta_t = r_{t+1} + \gamma v(s_{t+1}) - v(s_t)$ is called the TD error

Comparisons

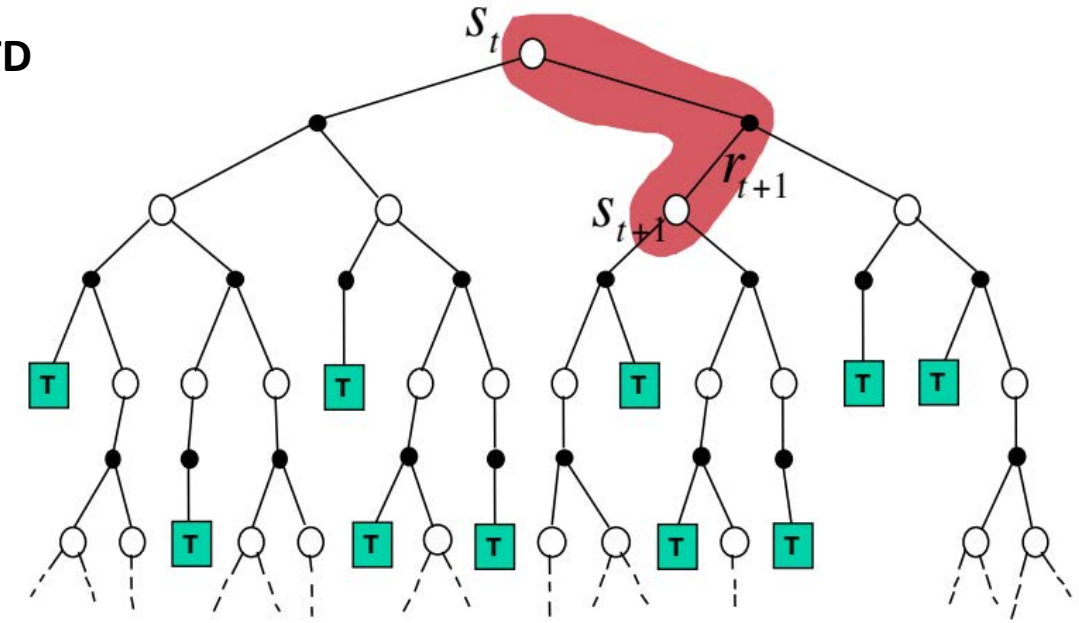
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

MC



$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

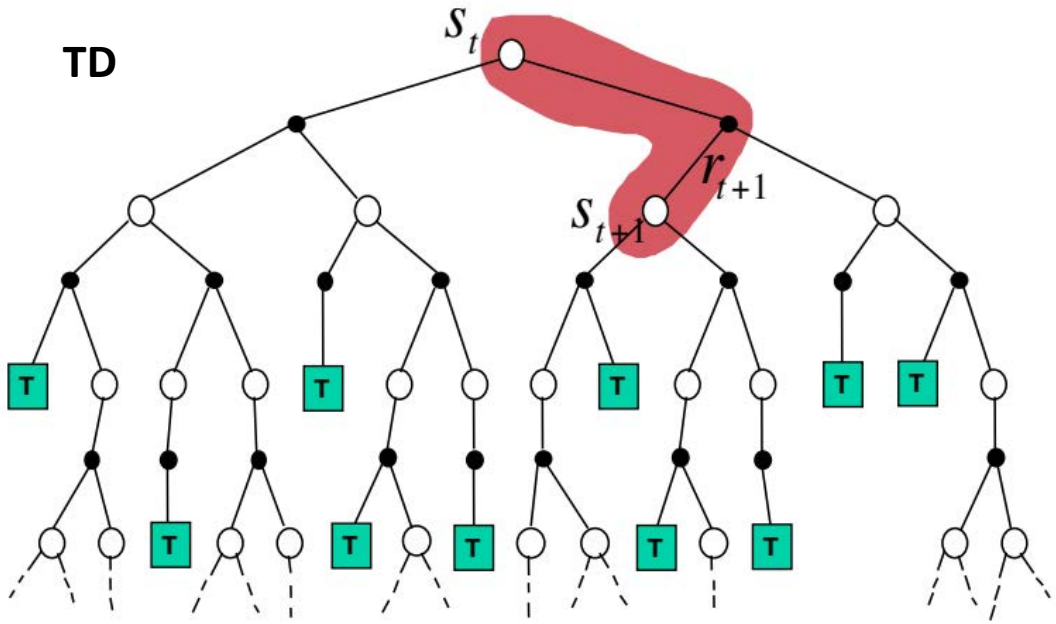
TD



Comparisons

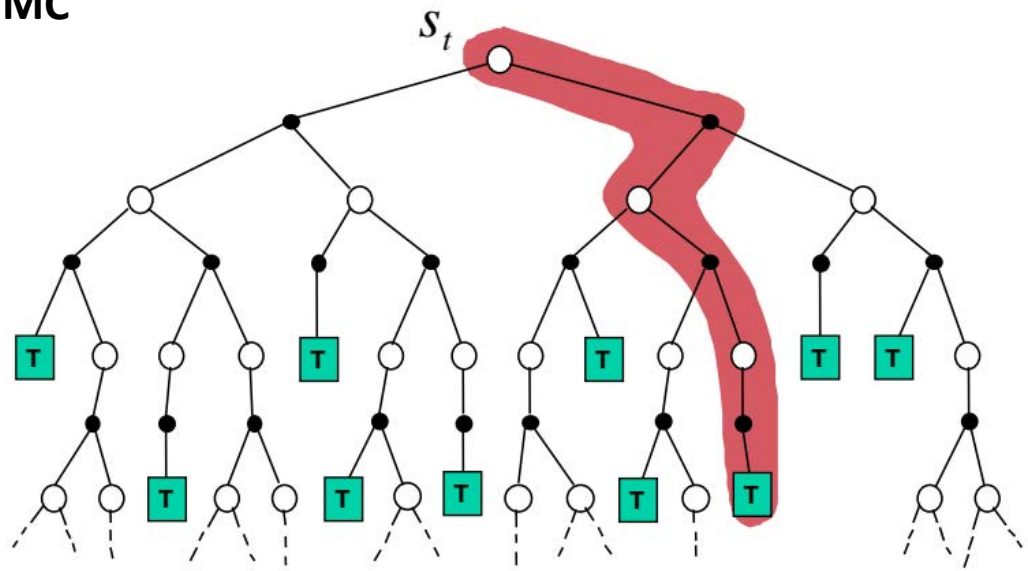
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

TD



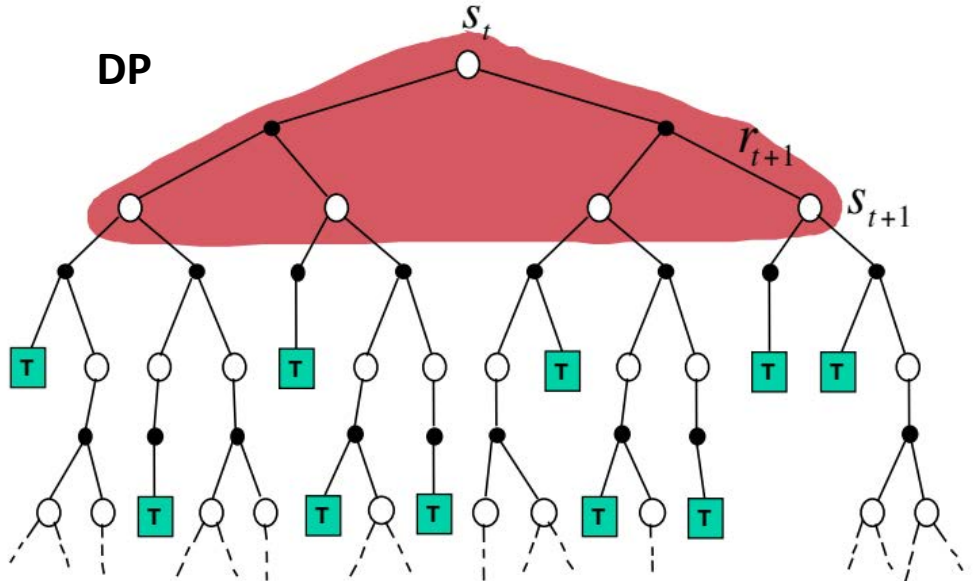
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

MC



$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$

DP



Policy Control

- Goal: Find the optimal policy
- One method: Find the optimal value function

  - Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi'(s) = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s, a)$$

- Two algorithms:
  - SARSA
  - Q-Learning

SARSA

- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

- On-Policy

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning

- Off-Policy

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$;
    until S is terminal

MC vs TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S; A)$
  - Use ϵ-greedy policy improvement
  - Update every time-step

Model-based Learning

- Use experience data to estimate model
- Compute optimal policy w.r.t the estimated model

Summary to RL

Planning	Policy evaluation	For a fixed policy	Value iteration, policy iteration
	Optimal control	Optimize Policy	
Model-free learning	Policy evaluation	For a fixed policy	Monte-carlo, TD learning, SARSA, Q-Learning
	Optimal control	Optimize Policy	
Model-based learning			

Large Scale RL

- So far we have represented value function by a lookup table
  - Every state s has an entry $v(s)$
  - Or every state-action pair s, a has an entry $q(s, a)$
- Problem with large MDPs:
  - Too many states and/or actions to sore in memory
  - Too slow to learn the value of each state (action pair) individually
  - Backgammon: 10^{20} states
  - Go: 10^{170} states

Solution: Function Approximation for RL

- Estimate value function with function approximation
  - $\hat{v}(s;\theta) \approx v_\pi(s)$ or $\hat{q}(s,a;\theta) \approx q_\pi(s,a)$
  - Generalize from seen states to unseen states
  - Update parameter θ using MC or TD learning
- Policy function
- Model transition function

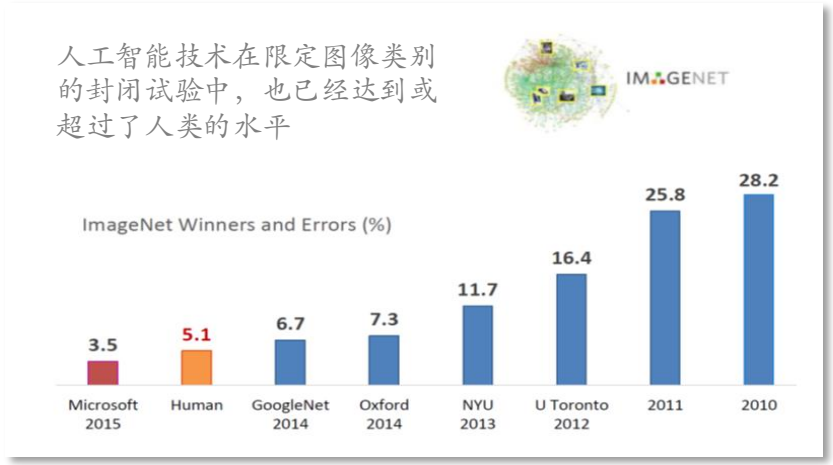
Deep Reinforcement Learning

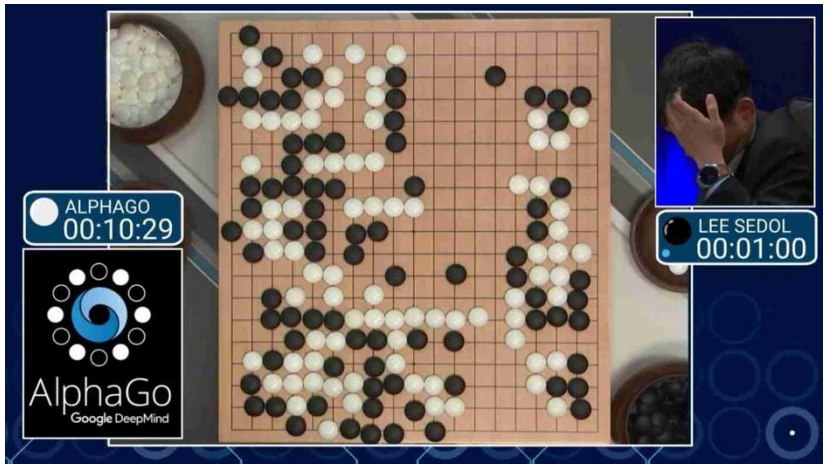
Deep learning . Value based . Policy gradients
Actor-critic . Model based

Deep Learning Is Making Break-through!

2016年10月，微软的语音识别系统在日常对话数据上，达到了5.9%的单词错误率，首次取得与人类相当的识别精度

人工智能技术在限定图像类别的封闭试验中，也已经达到或超过了人类的水平





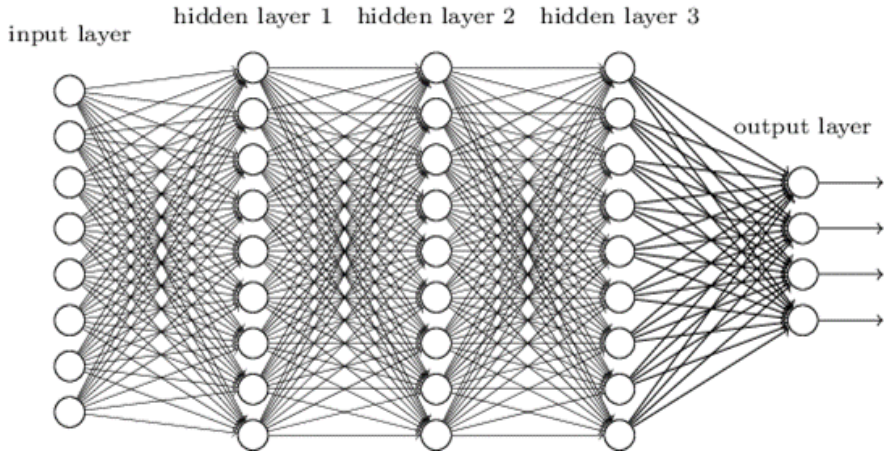
机器翻译新突破，微软中英新闻翻译达人类水平

原创 2018-03-15 camel AI科技评论

翻译没有唯一标准答案，它更像是一种艺术。

AI科技评论消息： 14 日晚，微软亚洲研究院与雷德蒙研究院的研究人员宣布，其研发的机器翻译系统在通用新闻报道测试集newstest2017 的中-英测试集上，达到了可与人工翻译媲美的水平；这是首个在新闻报道的翻译质量和准确率上可以比肩人工翻译的翻译系统。

Deep Learning



Deep learning (*deep machine learning*, or *deep structured learning*, or *hierarchical learning*, or sometimes *DL*) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using model architectures, with complex structures or otherwise, composed of multiple non-linear transformations.

1958: Birth of Perceptron and neural networks

1974: Backpropagation

Late 1980s: convolution neural networks (CNN) and recurrent neural networks (RNN) trained using backpropagation

1997: LSTM-RNN

2006: Unsupervised pretraining for deep neutral networks

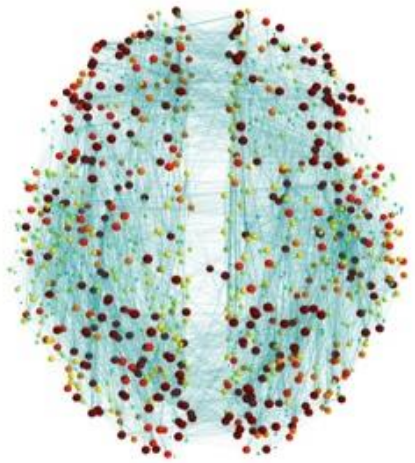
2012: Distributed deep learning (e.g., Google Brain)

2013: DQN for deep reinforcement learning

2015: Open source tools: MxNet, TensorFlow, CNTK

t

Driving Power

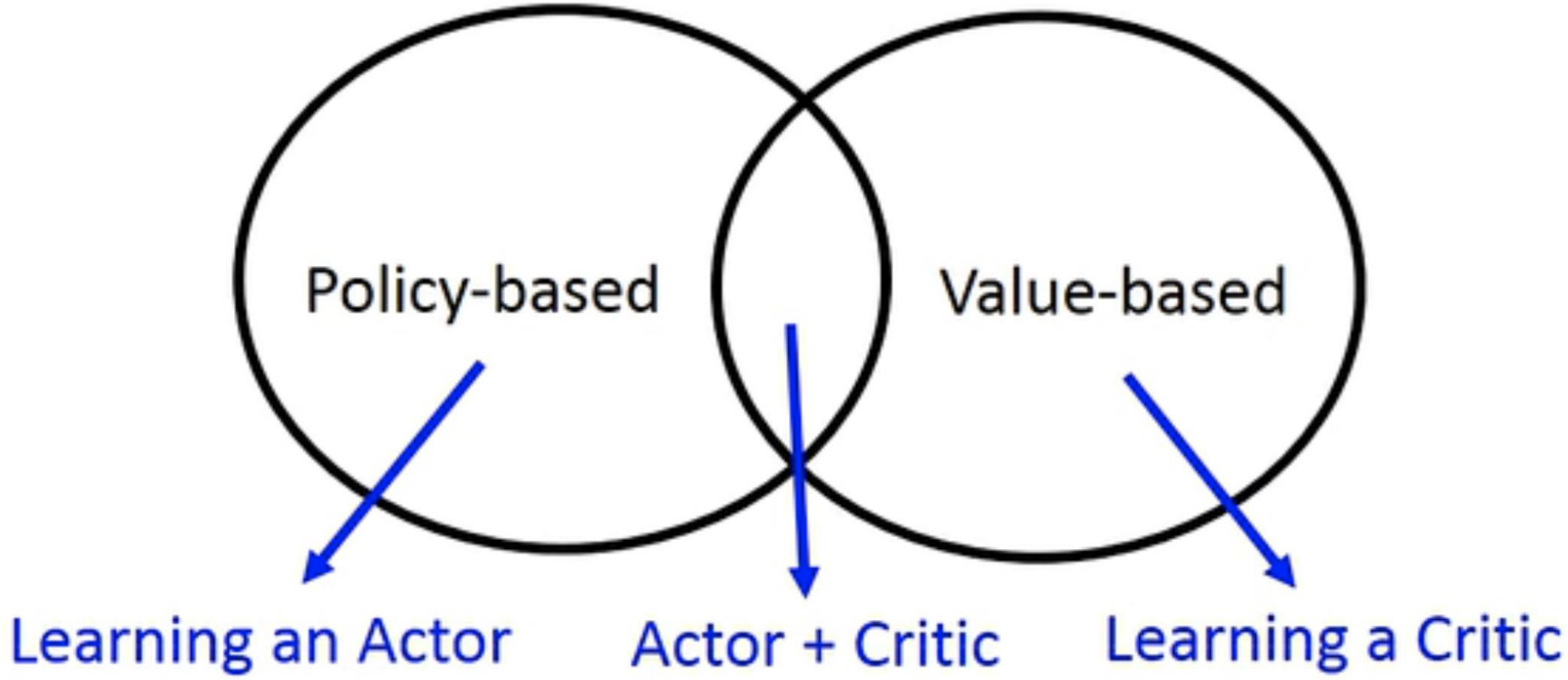


- Deep models: 1000+ layers, tens of billions of parameters

- Big computer clusters: CPU clusters, GPU clusters, FPGA farms, provided by Amazon, Azure, etc.

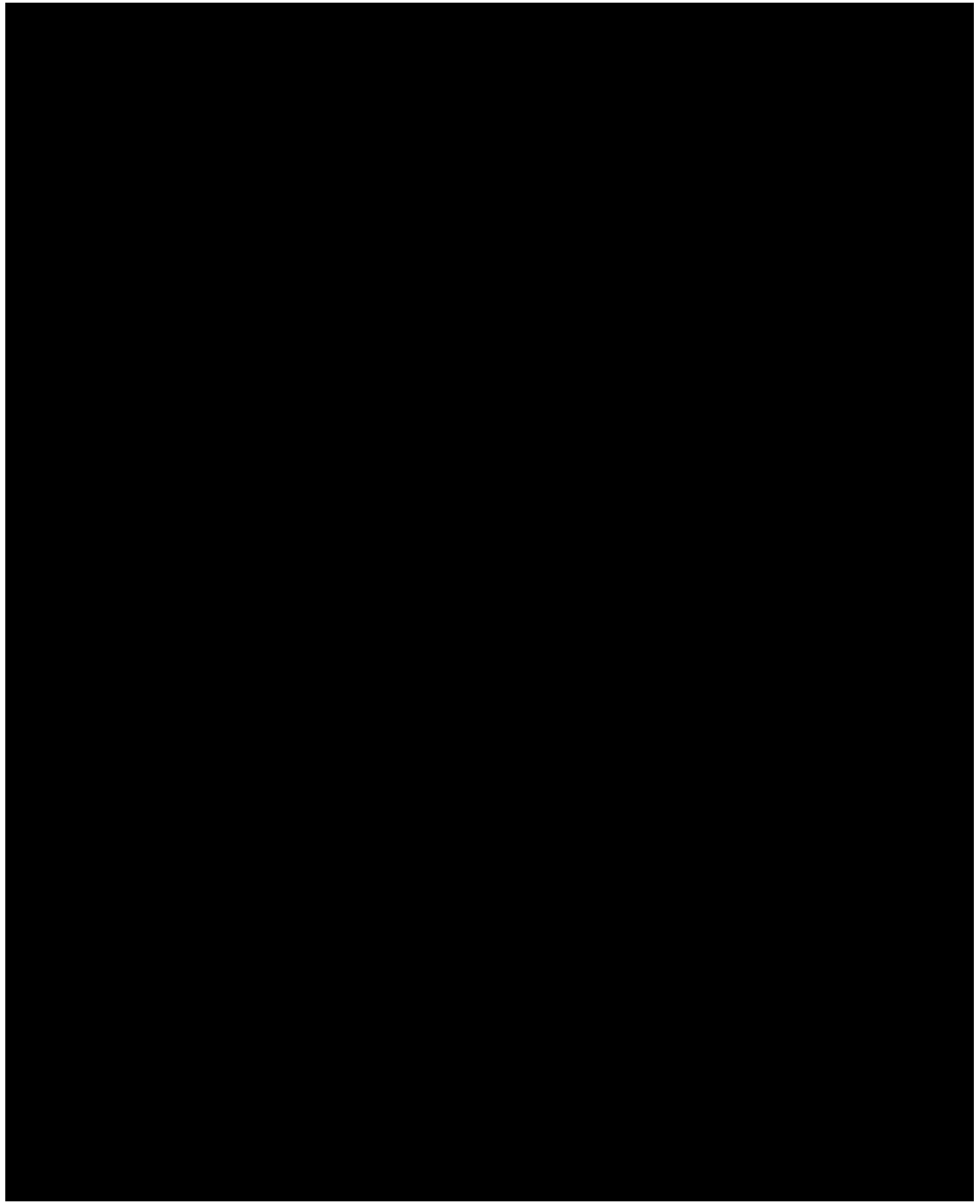
- Big data: web pages, search logs, social networks, and new mechanisms for data collection: conversation and crowdsourcing

Three Major Method



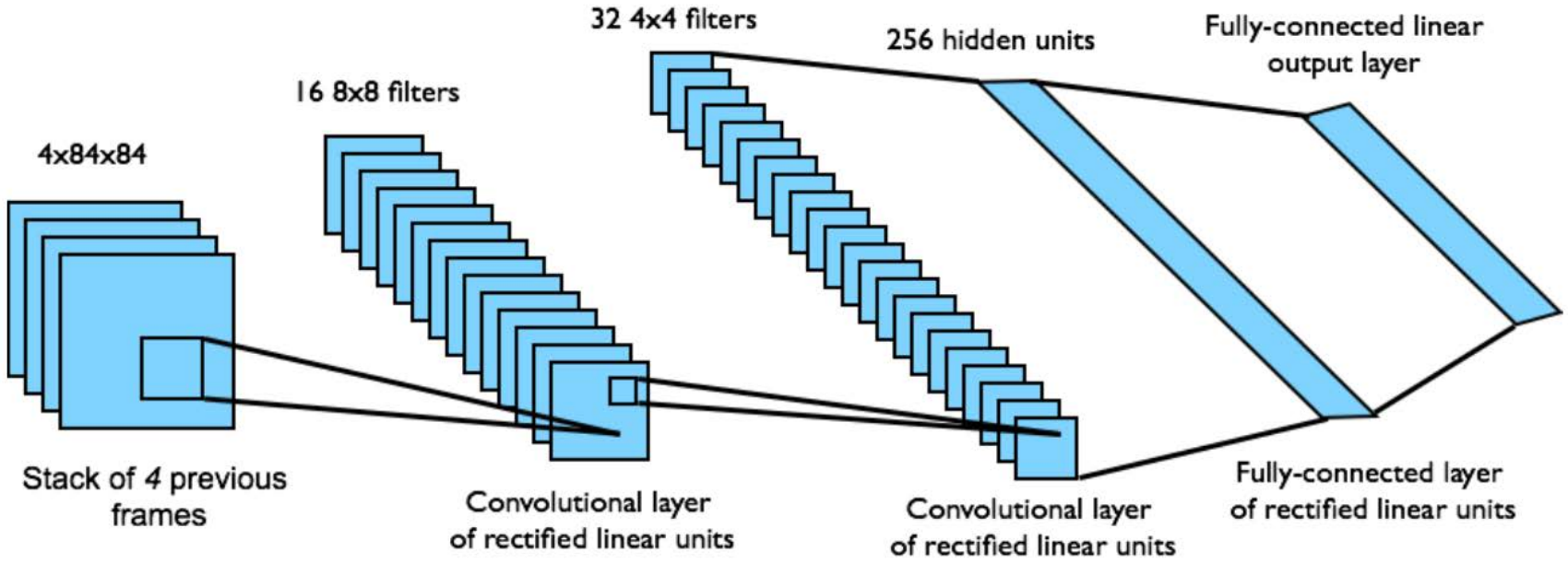
Value based methods: estimate value function or Q-function of the optimal policy (no explicit policy)

Nature 2015
Human Level Control Through Deep Reinforcement Learning



Representations of Atari Games

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Value Iteration with Q-Learning

- Represent value function by deep Q-network with weights θ

$$Q(s, a; \theta) \approx Q^{\pi}(s, a)$$

- Define objective function by mean-squared error in Q-values

$$L(\theta) = E\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right)^2\right]$$

- Leading to the following Q-learning gradient

$$\frac{\partial L(\theta)}{\partial \theta} = E\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right) \frac{\partial Q(s, a; \theta)}{\partial \theta}\right]$$

- Optimize objective end-to-end by SGD

Stability Issues with Deep RL

Naive Q-learning oscillates or diverges with neural nets

- Data is sequential
  - Successive samples are correlated, non-iid

- Policy changes rapidly with slight changes to Q-values
  - Policy may oscillate
  - Distribution of data can swing from one extreme to another

Deep Q-Networks

- DQN provides a stable solution to deep value-based RL
- Use experience replay
  - Break correlations in data, bring us back to iid setting
  - Learn from all past policies
  - Using off-policy Q-learning
- Freeze target Q-network
  - Avoid oscillations
  - Break correlations between Q-network and target

Deep Q-Networks: Experience Replay

To remove correlations, build data-set from agent's own experience

- Take action at according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D
- Sample random mini-batch of transitions (s, a, r, s') from D
- Optimize MSE between Q-network and Q-learning targets, e.g.

$$L(\theta) = E_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right)^2\right]$$

Deep Q-Networks: Fixed target network

To avoid oscillations, fix parameters used in Q-learning target

- Compute Q-learning targets w.r.t. old, fixed parameters θ^-

$$r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

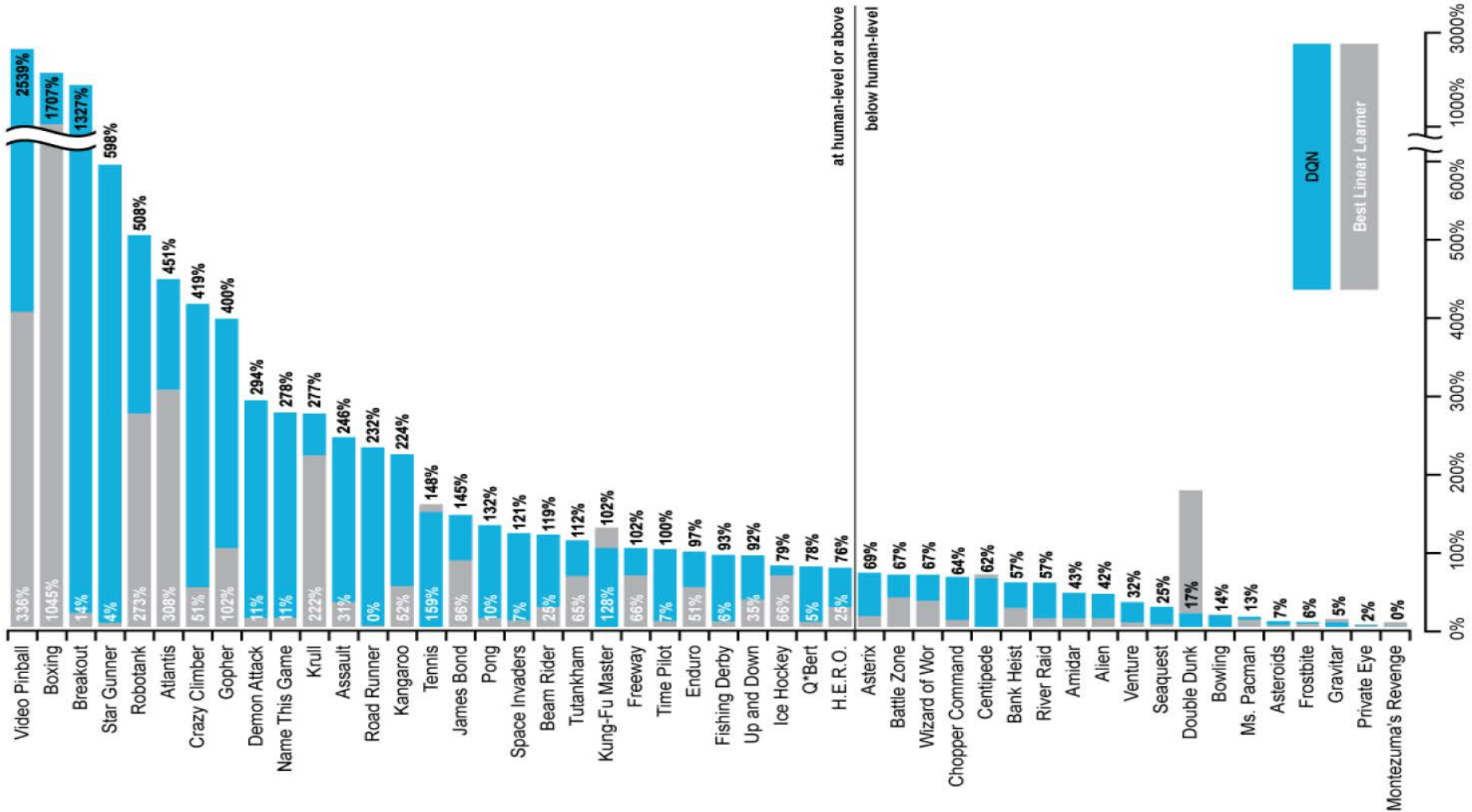
- Optimize MSE between Q-network and Q-learning targets

$$L(\theta) = E_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

- Periodically update fixed parameters $\theta^- \leftarrow \theta$

Experiment

Of 49 Atari games
43 games are better than state-of-art results
29 games achieves 75% expert score



	Q-learning	Q-learning + Target Q	Q-learning + Replay	Q-learning + Replay + Target Q
Breakout	3	10	241	**317**
Enduro	29	142	831	**1006**
River Raid	1453	2868	4103	**7447**
Seaquest	276	1003	823	**2894**
Space Invaders	302	373	826	**1089**

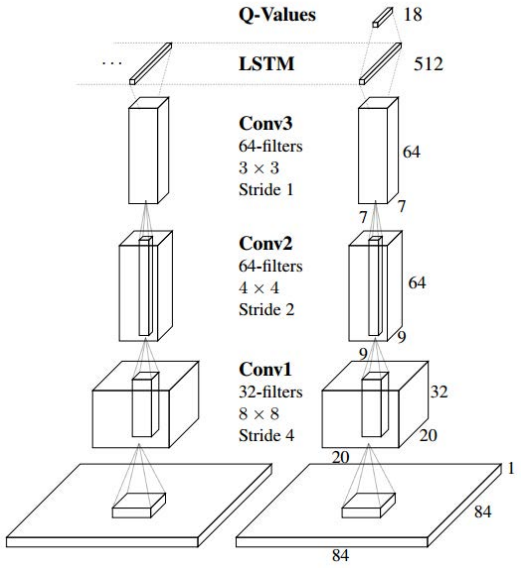
Other Tricks

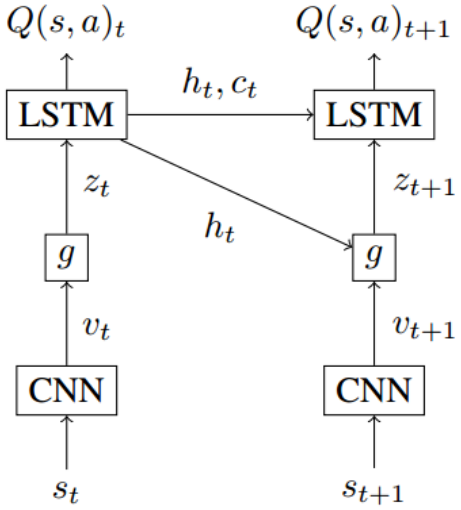
- DQN clips the rewards to [-1; +1]
- This prevents Q-values from becoming too large
- Ensures gradients are well-conditioned

- Can’t tell difference between small and large rewards
- Better approach: normalize network output
- e.g. via batch normalization

Extensions

- Deep Recurrent Q-Learning for Partially Observable MDPs
  - Use CNN + LSTM instead of CNN to encode frames of images
- Deep Attention Recurrent Q-Network
  - Use CNN + LSTM + Attention model to encode frames of images





Policy gradients: directly differentiate the objective

Gradient Computation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Policy Gradients

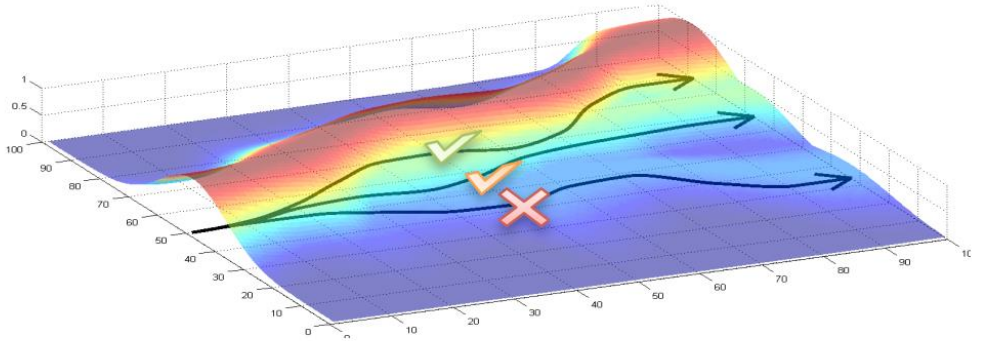
- Optimization Problem: Find θ that maximizes expected total reward.
  - The gradient of a stochastic policy $\pi_\theta(a|s)$ is given by

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]$$

  - The gradient of a deterministic policy $a = \mu_\theta(s)$ is given by

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]$$

- Gradient tries to
  - Increase probability of paths with positive R
  - Decrease probability of paths with negative R



REINFORCE

- We use return v_t as an unbiased sample of Q.
  - $v_t = r_1 + r_2 + \cdots + r_t$

function REINFORCE
Initialise θ arbitrarily
for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
for $t = 1$ to $T - 1$ **do**
$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
end for
end for
return θ
end function

- high variance
- limited for stochastic case

Actor-critic: estimate value function or Q-function of the current policy, use it to improve policy

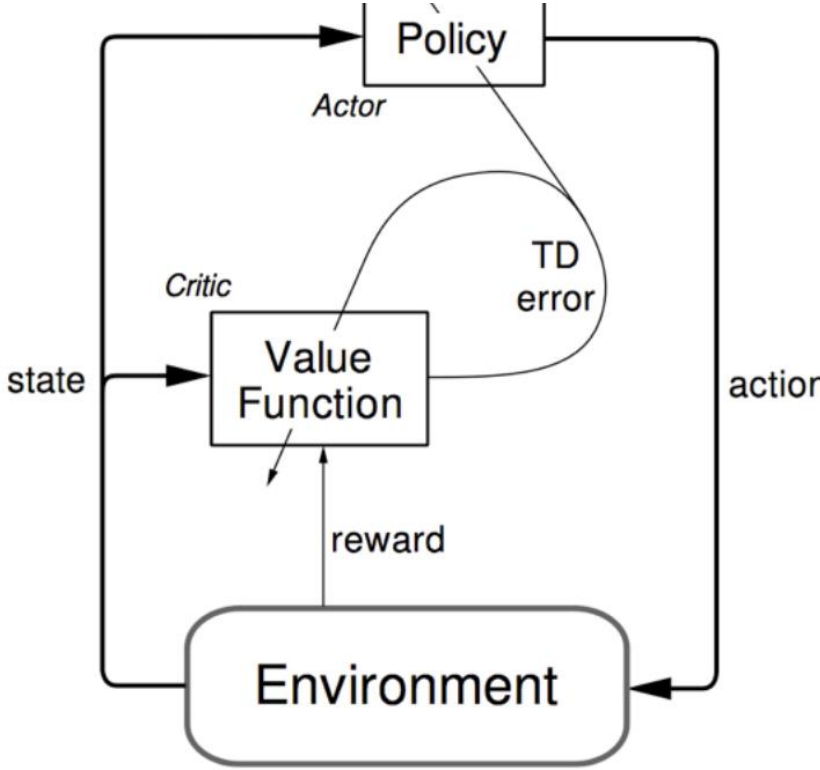
Actor-Critic

- We use a critic to estimate the action-value function

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$

- Actor-critic algorithms
  - Updates action-value function parameters
  - Updates policy parameters θ,
    in direction suggested by critic

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\}$
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function



Review

- Value Based
  - Learnt Value Function
  - Implicit policy
    - (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy

Model based DRL

- Learn a transition model of the environment/system

$$P(r, s'|s, a)$$

  - Using deep network to represent the model
  - Define loss function for the model
  - Optimize the loss by SGD or its variants
- Plan using the transition model
  - E.g., lookahead using the transition model to find optimal actions

Model based DRL: Challenges

- Errors in the transition model compound over the trajectory
- By the end of a long trajectory, rewards can be totally wrong
- Model-based RL has failed in Atari

Advanced topics in RL

Advanced topics in RL

- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- Learning from demonstrations
- Large Decision Model（LDM）
- RL for LLM
- LLM and agent

Recap: Deep Q-learning

- DQN: use deep neural networks to perform Q-learning

- Two major innovations
  - Experience replay
  - Freeze target network

$(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D

$$L(\theta) = E_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

- Idea: make RL look more like supervised learning
  - Break correlations in data
  - Break correlations between Q-network and target

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

Prioritized replay

- DQN samples uniformly from replay
  - However, different samples are **not equally** important
- Idea: prioritize transitions on which we can learn much!
- Sample experience according to **priority function**

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

DQN error

$$P(i) = \frac{p_i^\beta}{\sum_k p_k^\beta}$$

priority

Priority of a tuple (s_i, a_i, r_{I+1}, s_i) is proportional to its DQN error

Schaul, Tom, et al. "Prioritized Experience Replay." ICLR (Poster). 2016.

Double DQN

- Double Q-Learning: maintain 2 Q-functions Q_1, Q_2
  - Randomly choose $Q_1(Q_2)$ to **select action** with 0.5 probability
  - Choose the other $Q_2(Q_1)$ to **evaluate action**
- Double DQN: extend the above idea to DQN
  - **Current Q-network $\boldsymbol{w}$ is used to select actions**
  - **Older Q-network $\boldsymbol{w}^-$** is used to evaluate actions

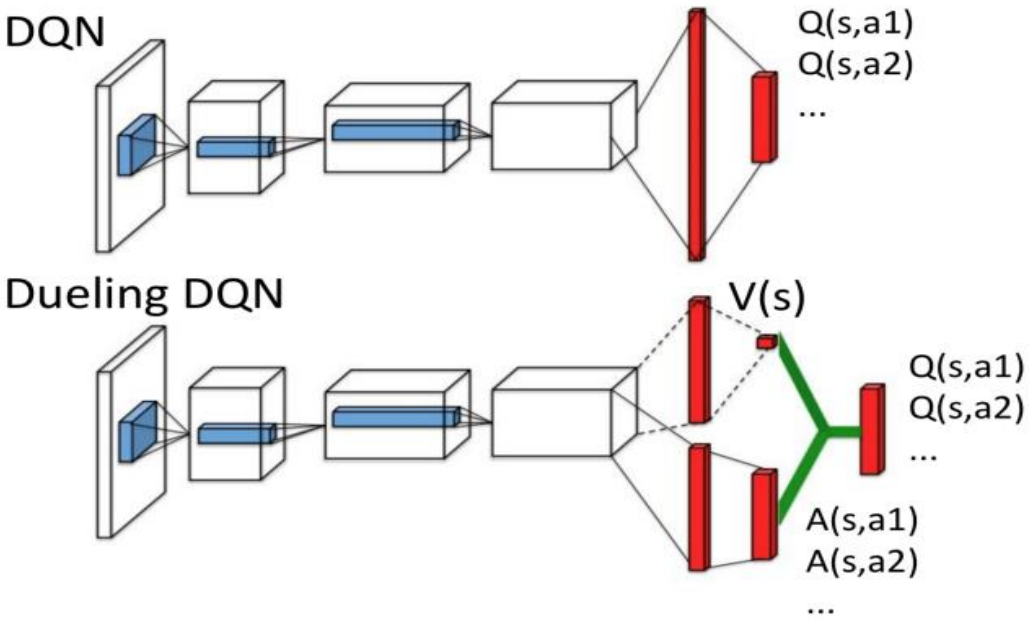
$$\Delta \boldsymbol{w} = \alpha\Big(r + \gamma \overbrace{\hat{Q}\big(\underbrace{\arg\max_{a'} \hat{Q}(s', a'; \boldsymbol{w})}_{\text{Action selection: } \boldsymbol{w}}; \boldsymbol{w}^-\big)}^{\text{Action evaluation: } \boldsymbol{w}^-} - \hat{Q}(s, a; \boldsymbol{w})\Big)$$

Break the correlation between Q-network and target

Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. No. 1. 2016.

Dueling networks

- Intuition: Features needed to accurately represent value may be different than those needed to specify difference in actions
  - Game score may help accurately predict $V(s)$
  - But not necessarily in indicating relative action values $Q(s, a_1)$ v.s $Q(s, a_2)$
- Idea: decompose $q_\theta(s, a) = v_\varphi(s) + A_\omega(s, a)$, where $\theta = \varphi \cup \omega$



$v_\varphi(s)$: general state value

$A_\omega(s, a)$: advatange of taking action $\boldsymbol{a}$

Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *International conference on machine learning*. PMLR, 2016.

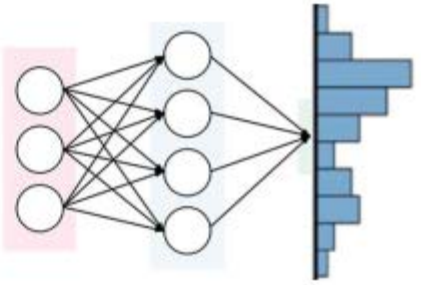
Distributional RL

- So far, we study RL in the form of **expectation**, e.g. expected returns
- We could also move towards learning the **distribution** of returns
- Why learning distribution?
  - Suppose two actions with the **same expected return**
  - The one with lower **variance** is more **stable**

Distributions supported on a finite support $\{z_1, \ldots, z_n\}$

Discrete distribution $\{p_i(x, a)\}_{1 \leq i \leq n}$

$$Z(x, a) = \sum_i p_i(x, a)\delta_{z_i}$$



Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *International Conference on Machine Learning*. PMLR, 2017.

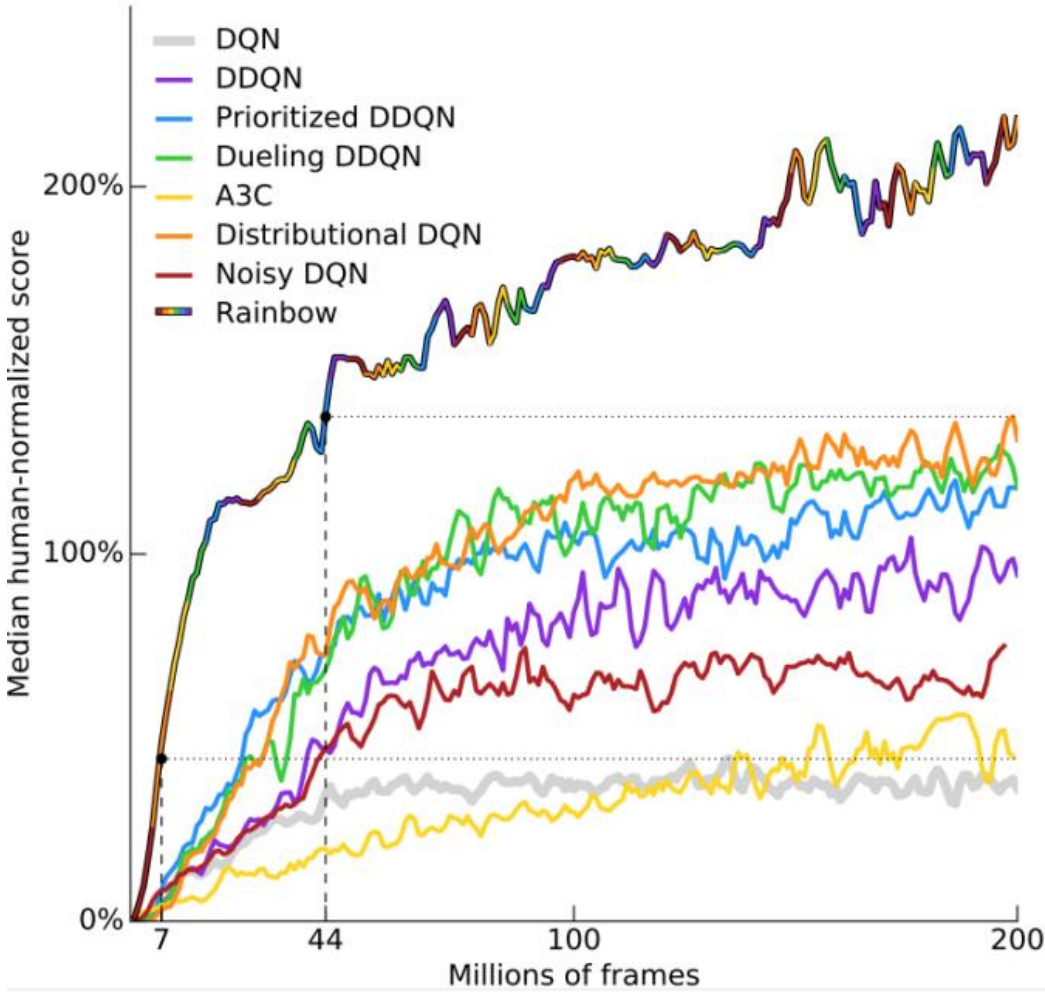
Rainbow DQN

- Put all the above improvements together!
  - Double DQN
  - Prioritized replay
  - Dueling networks
  - Distributional DQN
  - Multi-step learning
  - Noisy DQN

covered in previous pages

not covered in this lecture

- Very competitive against other algorithms using only one strategy



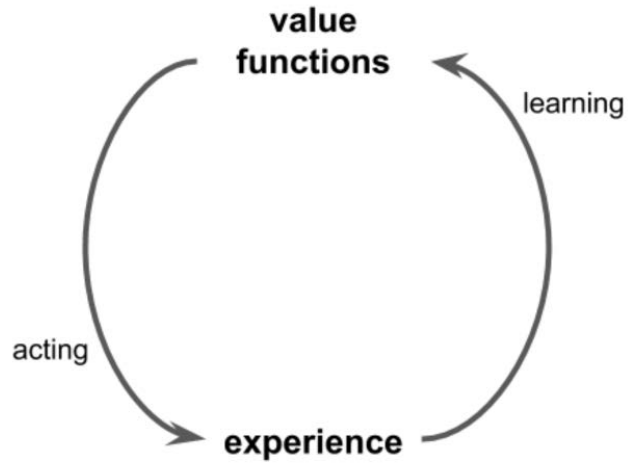
Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." *Thirty-second AAAI conference on artificial intelligence*. 2018.

Advanced topics in RL

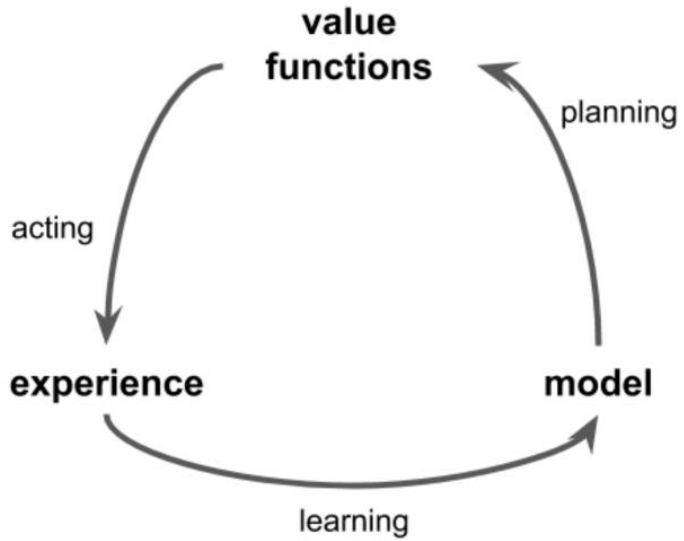
- Improvements on DQN
- **Bridging the gap between model-free and model-based RL**
- Offline RL
- Learning from demonstrations
- Large Decision Model（LDM）
- RL for LLM
- LLM and agent

Recap: model-free & model-based RL

- Model-free RL
  - No model
  - Learn value functions from experience



- Model-based RL
  - Learn a model from experience
  - Plan value functions using the learned model



Combining model-free & model-based RL

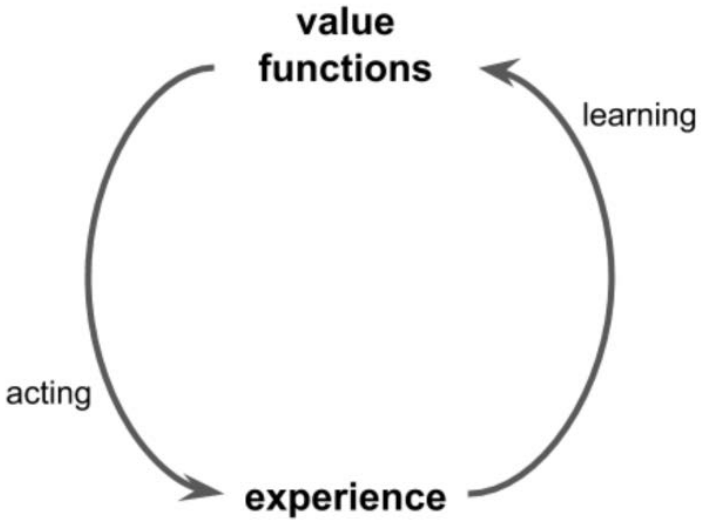
- Model-free RL: interacting with the environment can be **expensive**
- Model-based RL: the learned model can be **inaccurate**
- Solution: combining them in a single algorithm!

- We consider two sources of experience
  - **Real experience:** sampled from environment (true MDP)
  - **Simulated experience:** sampled from model (approximate MDP)
- Dyna: **integrate the two sources of experience**

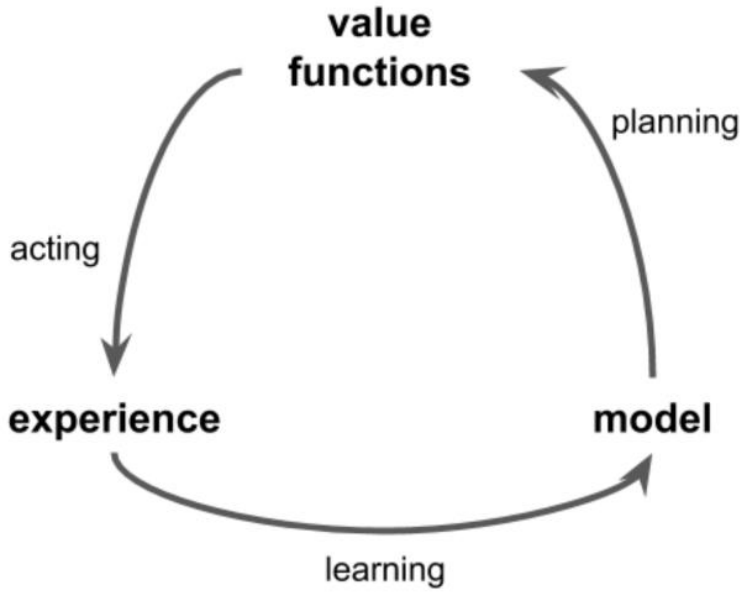
Sutton, Richard S. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming." *Machine learning proceedings 1990*. Morgan Kaufmann, 1990. 216-224.

Dyna

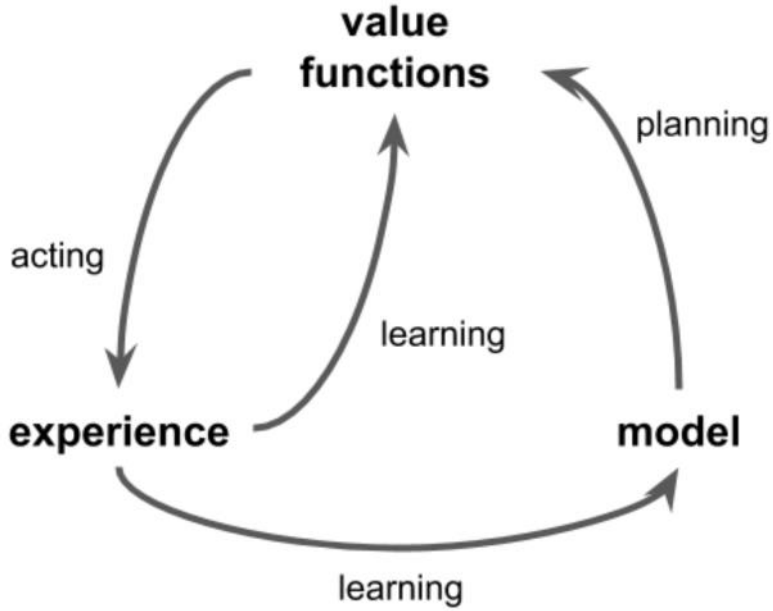
- Learn a model from real experience
- Learn AND plan value function (policy) from real and simulated experience
- Treat real and simulated experience equivalently.



model-free



model-based



Dyna

Dyna-Q algorithm

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Do forever:

(a) $s \leftarrow$ current (nonterminal) state

(b) $a \leftarrow \varepsilon$-greedy(s,Q)

(c) Execute action a; observe resultant state, s', and reward, r

(d) $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

(e) $Model(s,a) \leftarrow s', r$ (assuming deterministic environment)

(f) Repeat N times:

$s \leftarrow$ random previously observed state

$a \leftarrow$ random action previously taken in s

$s', r \leftarrow Model(s,a)$

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

model-free learning

model-based planning

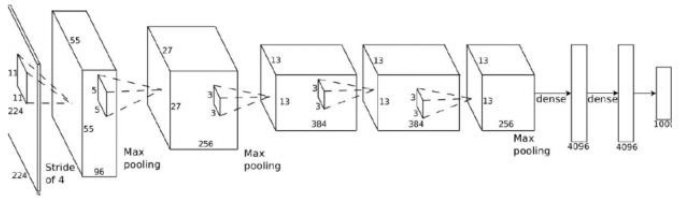
Sutton, Richard S. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming." *Machine learning proceedings 1990*. Morgan Kaufmann, 1990. 216-224.

Advanced topics in RL

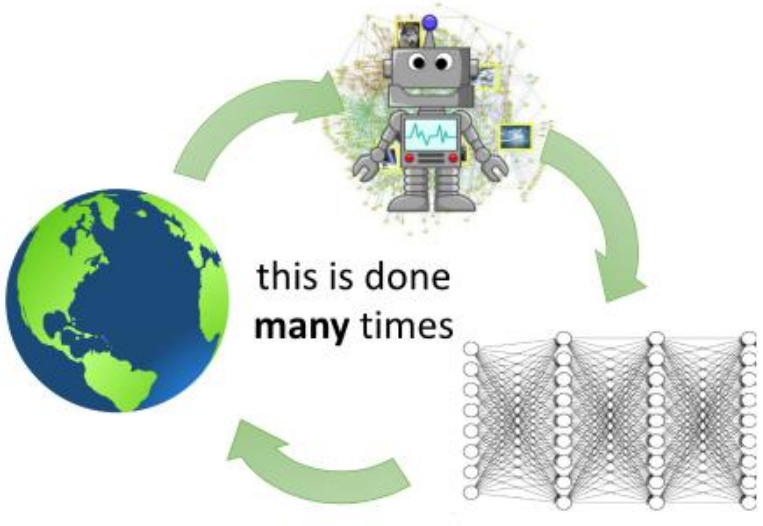
- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- Learning from demonstrations
- Large Decision Model（LDM）
- RL for LLM
- LLM and agent

Recap: SL and RL

- Supervised Learning: passive, data-driven

Big Data   Powerful Model

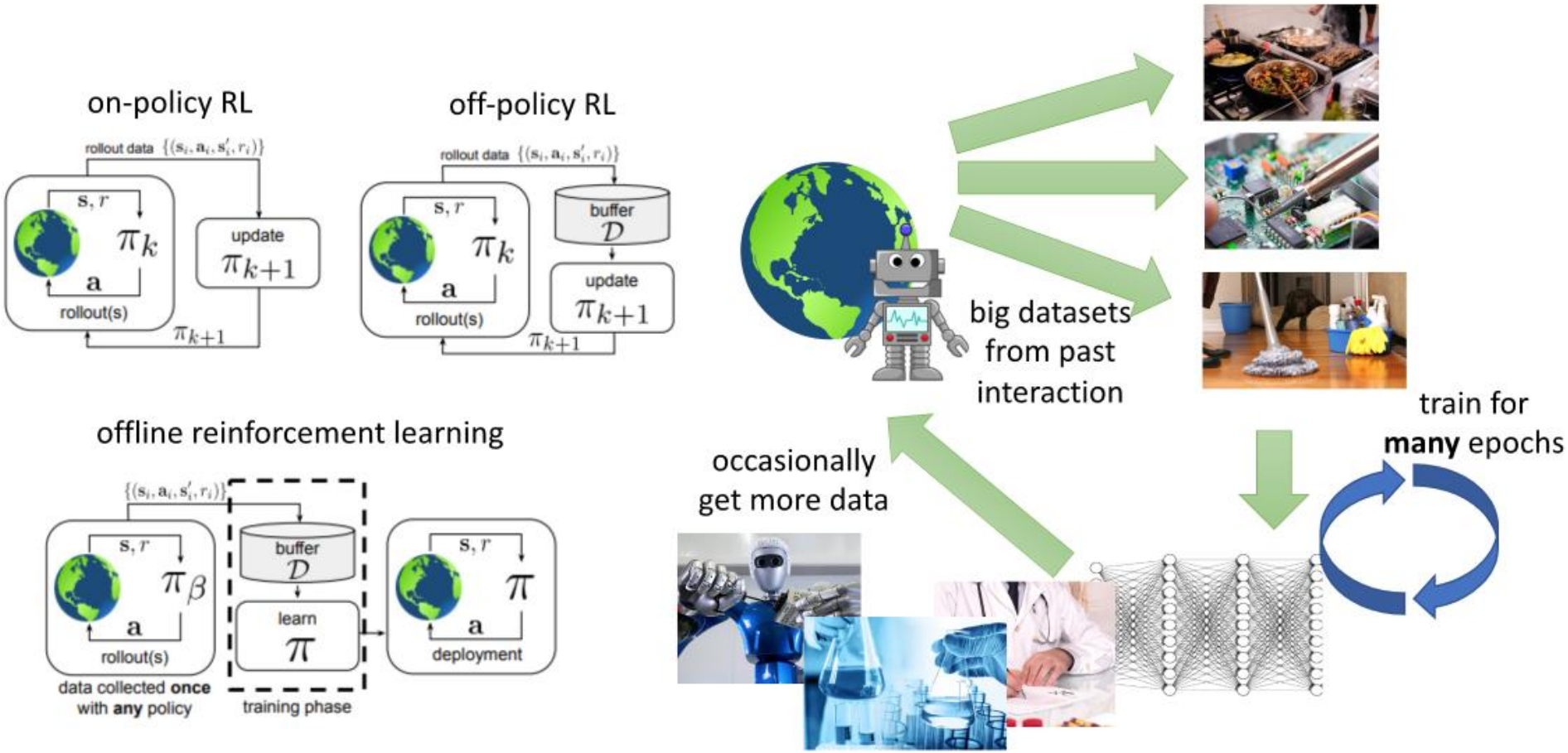
- Reinforcement Learning: active, interaction-driven



Your action determines the data you observe!

What is Offline RL

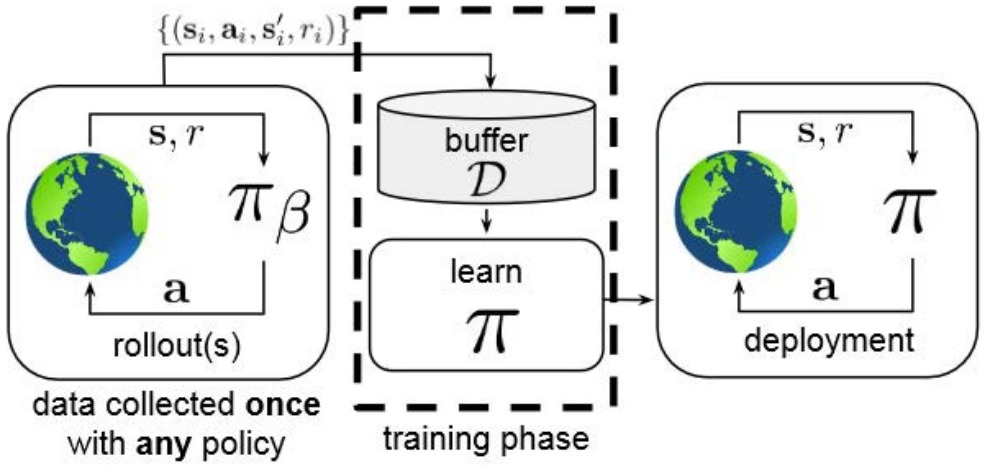
- Question: can we develop data-driven RL methods?



No need to interact with the environment!

What is Offline RL

- Learn a new policy π_θ,
- from data collected by a behavior policy π_β
- Goal: π_θ is much better than π_β



Formally:

$$\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$$

$$\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$$
$$\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$$ ← generally **not** known
$$\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$
$$r \leftarrow r(\mathbf{s}, \mathbf{a})$$

RL objective: $\max_\pi \sum_{t=0}^{T} E_{\mathbf{s}_t \sim d^\pi(\mathbf{s}), \mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s})}[\gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$

Why Offline RL

- Interacting with the environment (online RL) can be
  - Expensive
  - Risky
  - Complicated
  - Dangerous

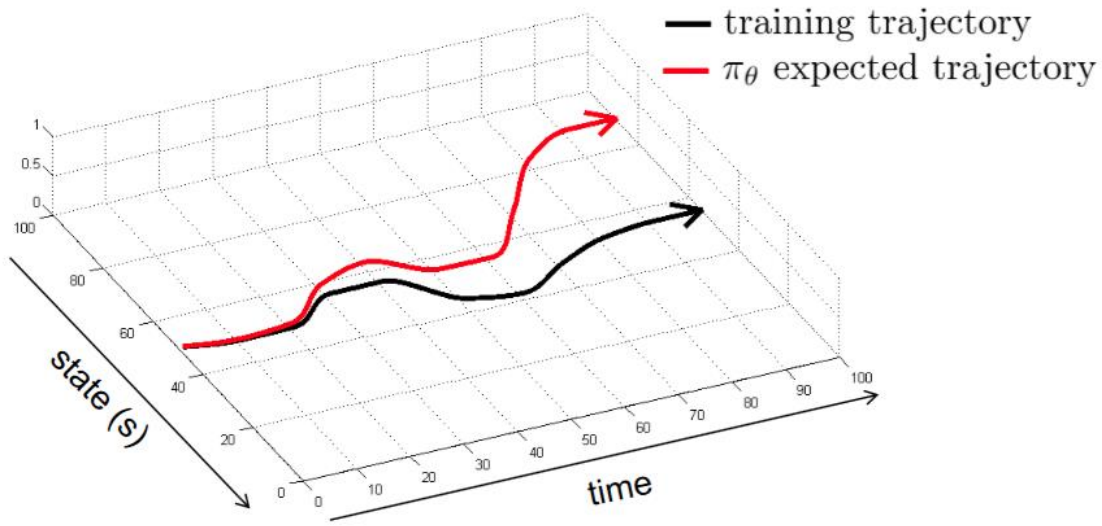
Major challenge: distribution shift

- The action of the learned policy π_θ can change what we see
- Particularly, π_θ may visit states that π_β rarely or never visit
- The state frequency $d(s)$ of π_θ can be very **different** from π_β
- Now we break the IID assumption, and the errors can be large

Simple example:

behavioral cloning
train π_θ to *copy*
assume data is *optimal*

error scales as $O(T^2)$



Idea: policy constraints

- We can regularize π_θ to be not too different from π_β

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\mathbf{a}'|\mathbf{s}')}[Q(\mathbf{s}', \mathbf{a}')]$$

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] \quad \text{s.t. } D(\pi_\theta(\mathbf{a}|\mathbf{s}), \pi_\beta(\mathbf{a}|\mathbf{s})) \leq \varepsilon$$

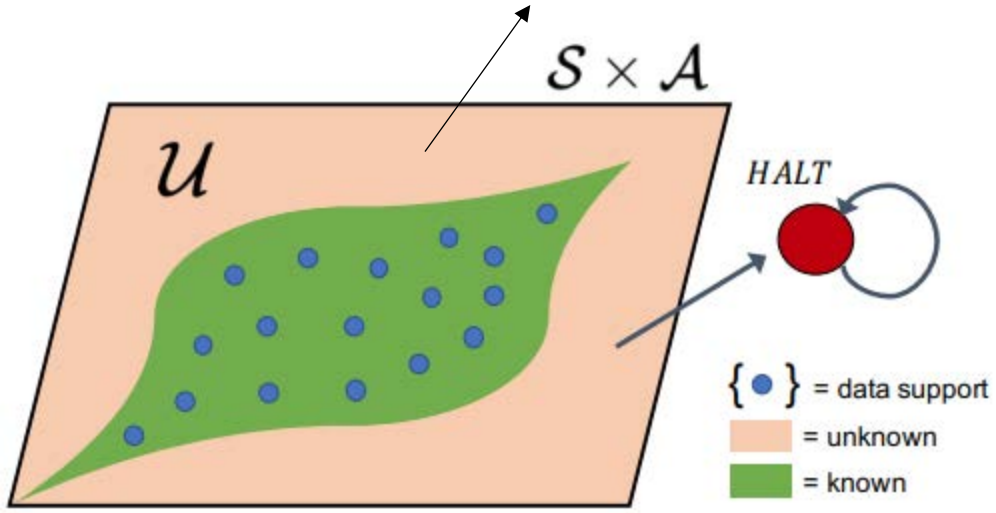
- Constraint options
  - KL divergence
  - f divergence
  - …

Kumar, Aviral, et al. "Stabilizing off-policy q-learning via bootstrapping error reduction." *Advances in Neural Information Processing Systems* 32 (2019).
Fujimoto, Scott, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration." International Conference on Machine Learning. PMLR, 2019.

Idea: conservative model-based RL

- We can learn a model that limits the reward of unknown scenarios
  - Keep unaltered in known (s, a) pairs
  - Make reward function conservative in unknown (s, a) pairs

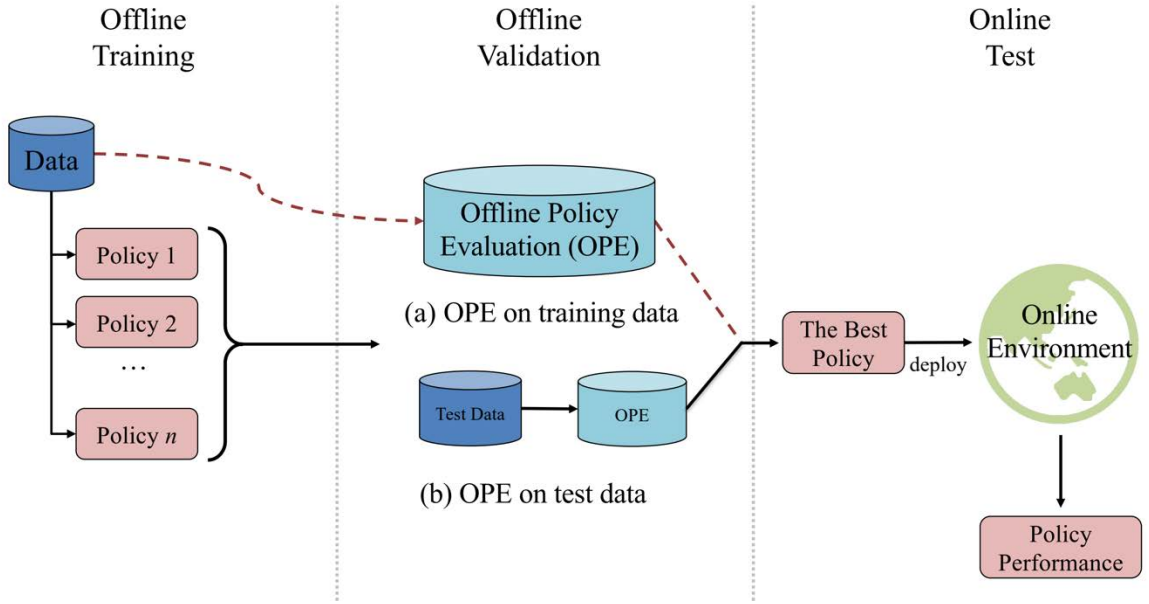
Force the reward to be low in unknown scenarios!



Kidambi, Rahul, et al. "Morel: Model-based offline reinforcement learning." *Advances in neural information processing systems* 33 (2020): 21810-21823.
Yu, Tianhe, et al. "Mopo: Model-based offline policy optimization." Advances in Neural Information Processing Systems 33 (2020): 14129-14142.

Evaluation

- We might still need a simulator/rollouts to see if our method works
- Use both offline and online evaluation



Offline RL is a very active area of research!

Advanced topics in RL

- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- **Learning from demonstrations**
- Large Decision Model（LDM）
- RL for LLM
- LLM and agent

Motivation

- So far, we assume the reward is given
- However, defining good reward functions can be difficult
- Or, we can implicitly specify them through **demonstrations**

Define reward manually ✗

Demonstrate good policy ✓

Imitation Learning

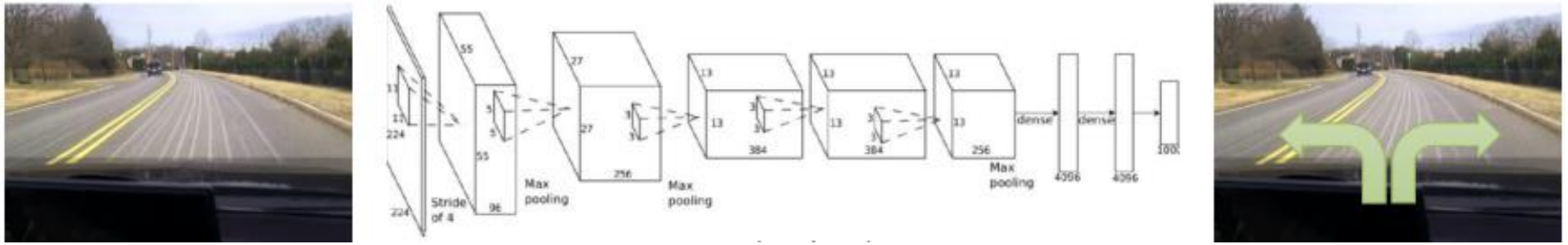
- **Expert** provides a set of **demonstration trajectories**: sequences of states and actions
- Imitation learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:
  - Specifying a reward that would generate such behavior,
  - Specifying the desired policy directly

Imitation Learning

- **Input**
  - state space, action space
  - (sometimes) transition model $P(s'|s,a)$
  - no reward function R
  - samples $\{\tau_i\}$ drawn from teacher's policy π^*
- Behavior Cloning
  - Directly learn the teacher's **policy** using **supervised learning**
- Inverse RL
  - Recover R
  - Use it to generate a good policy

Behavior Cloning

- Idea: directly copy the observed behaviors
  - Fix a policy class (e.g. neural networks)
  - Estimate a policy from training examples $(s_0, a_0), (s_1, a_1), \ldots$
- This is a standard supervised learning problem



s_t $\pi_\theta(a_t|s_t)$ a_t

Pomerleau, Dean A. "Alvinn: An autonomous land vehicle in a neural network." *Advances in neural information processing systems* 1 (1988).

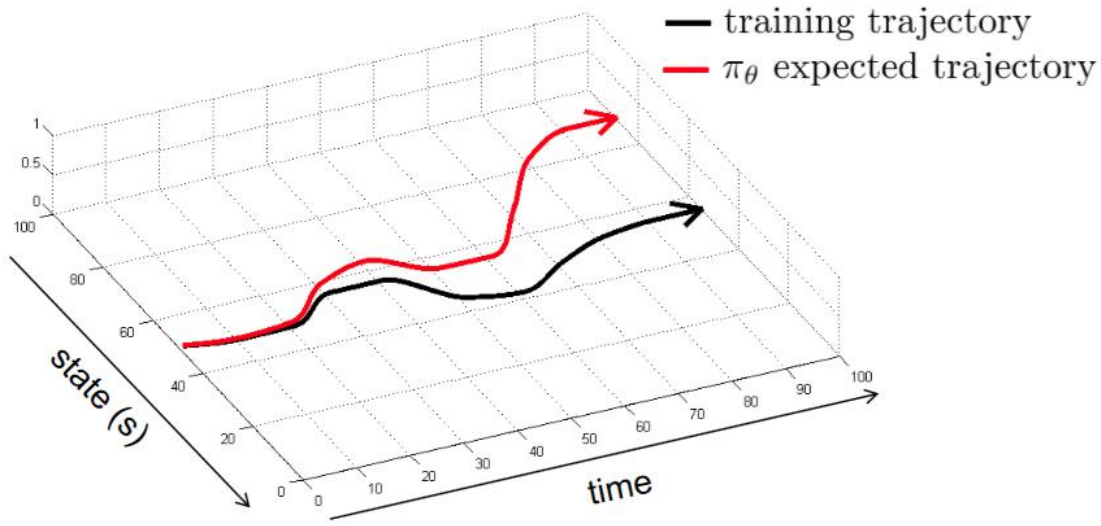
Behavior Cloning

- Limitation: the errors can be large
- Recap: behavior cloning is offline RL

Simple example:

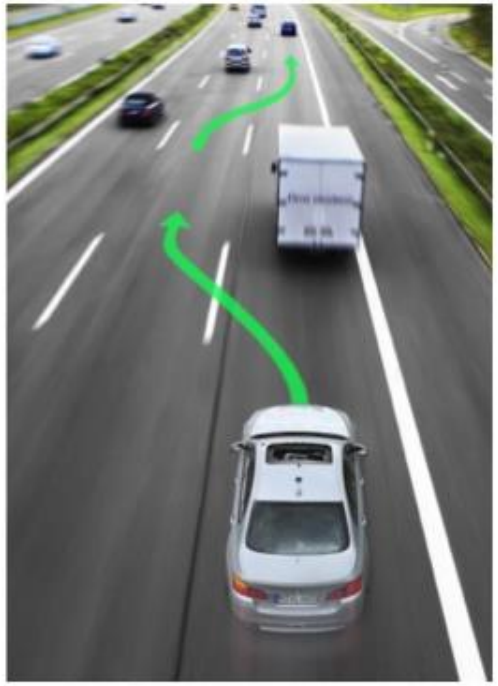
behavioral cloning
train π_θ to *copy*
assume data is *optimal*

error scales as $O(T^2)$

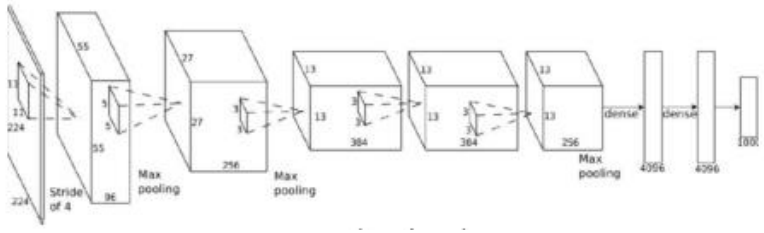


Inverse RL

- Idea: inferring rewards from observed behavior
- Learning a reward function such that the expert is (near) optimal
- Then use the reward function to learn a policy



$\boldsymbol{r(s,a)}$



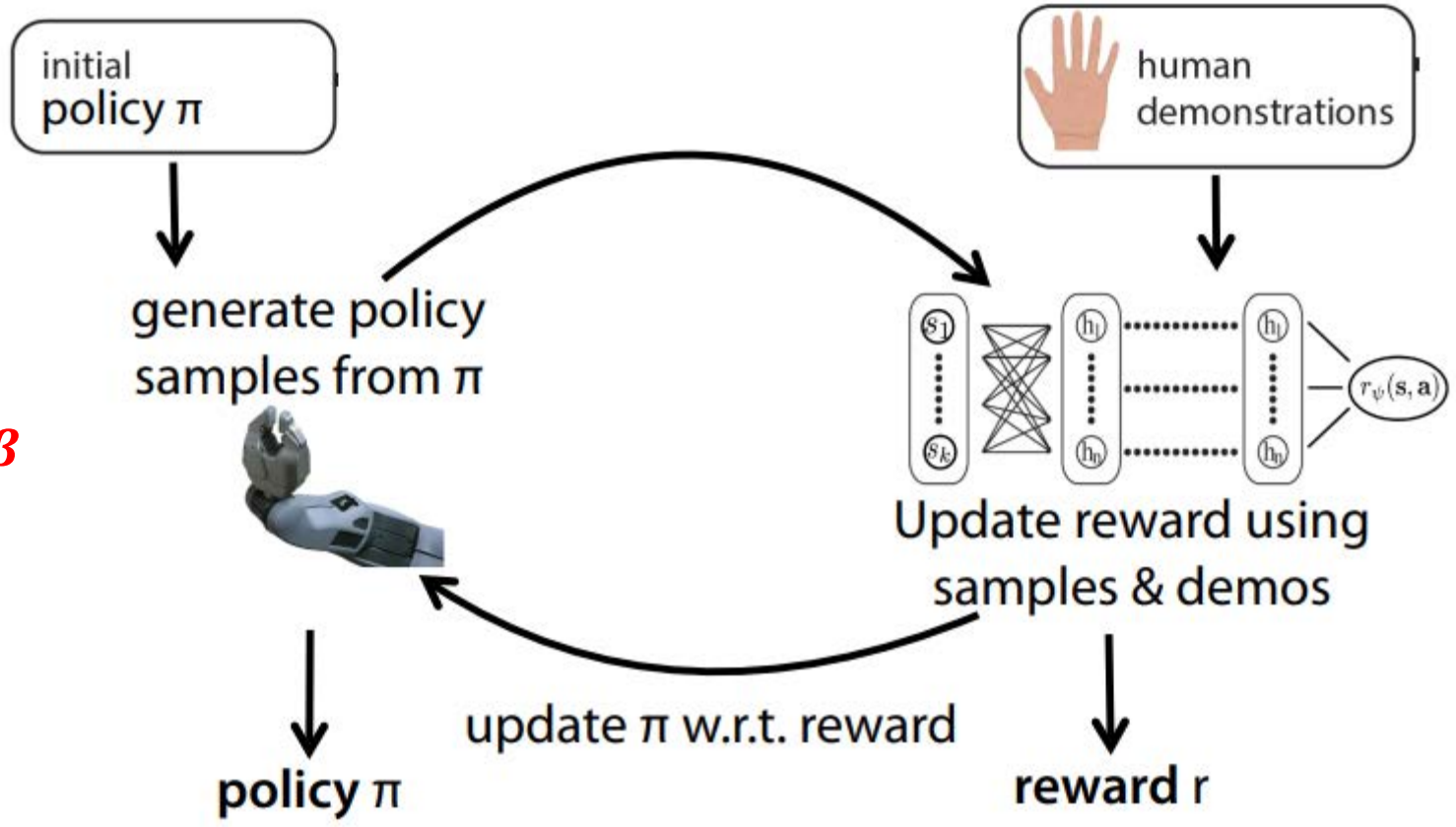
$\boldsymbol{\pi}$

Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." *ICML*. Vol. 1. 2000.

Inverse RL

- Example: Guided Cost Learning (Finn et al. ICML16)

Update $\boldsymbol{\pi_\theta}$ to be close to $\boldsymbol{\pi_\beta}$ w.r.t $\boldsymbol{r}$



Update $\boldsymbol{r}$ such that $\boldsymbol{\pi_\beta}$ is better $\boldsymbol{\pi_\theta}$

Finn, Chelsea, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization." *International conference on machine learning*. PMLR, 2016.

Advanced topics in RL

- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- Learning from demonstrations
- **Large Decision Model（LDM）**
- RL for LLM
- LLM and agent

Data Dependency of LDM

Data of Large Text/Vision Model:

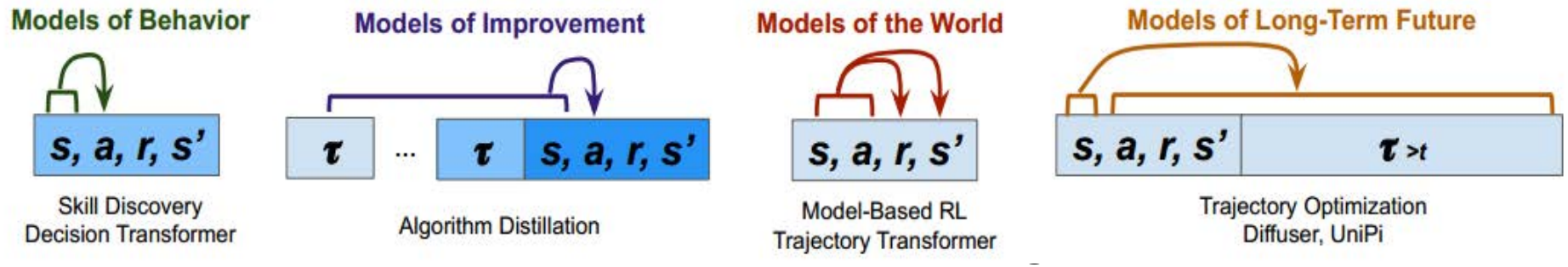
- Static Text/Image Records $x \sim \mathcal{D}$

Data of Large Decision Model:

- Interactive Decision Trajectories： $\tau \sim \mathcal{D}_{\mathrm{RL}}$
- Structure of Trajectories： State、Action、Reward：

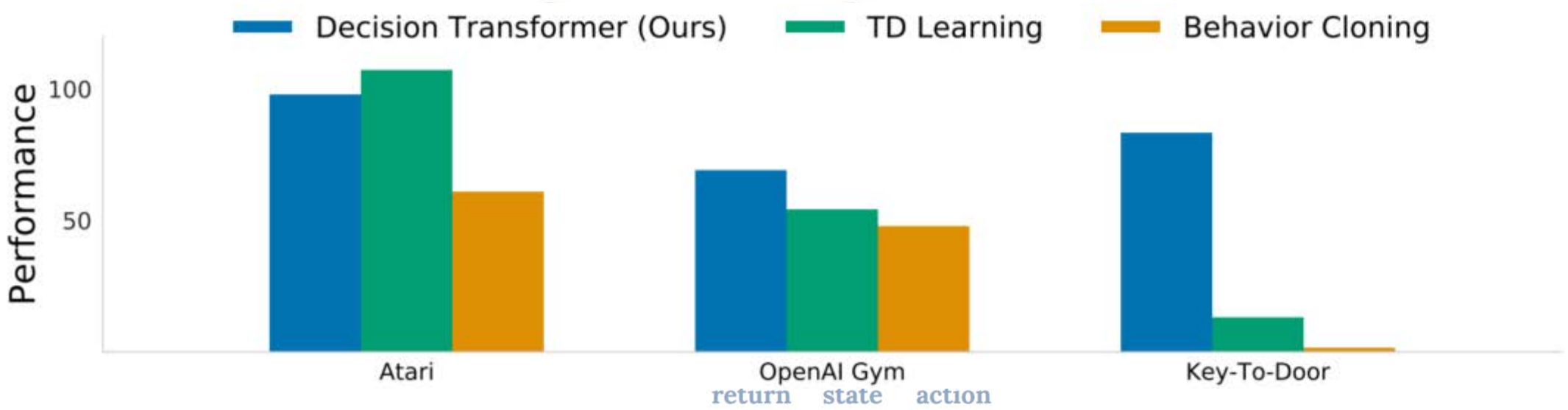
$$\tau := \{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_H, a_H, r_H)\}$$

- Trajectory under a Given Policy：



Literature of LDM: Action Modeling

- **Research Problem：** How to generate actions instead of traditional RL?
- **Algorithm Design： Decision Transformer**
  - Extract features via MLP/CNN from tokens and add timestep remarks
  - Generate action tokens via Transformer $\tau = \left(\widehat{R}_1, s_1, a_1, \widehat{R}_2, s_2, a_2, \ldots, \widehat{R}_T, s_T, a_T\right)$
  - Cross-Entropy Loss for Discrete Actions, MSE Loss for Continuous Actions



[1] Decision Transformer: Reinforcement Learning via Sequence Modeling. NeurIPS, 2021.

Literature of LDM: Environment Modeling

- **Research Problem：** How to generate transitions instead of traditional RL?
- **Algorithm Design：** **Trajectory Transformer**
  - Discretize Actions and States: $\boldsymbol{\tau} = (\ldots, s_t^1, s_t^2, \ldots, s_t^N, a_t^1, a_t^2, \ldots, a_t^M, r_t, \ldots)$
  - Compute Loss Along Reward, State and Action:

$$\mathcal{L}(\tau) = \sum_{t=1}^{T} \Big(\sum_{i=1}^{N} \log P_\theta\big(s_t^i \mid \mathbf{s}_t^{<i}, \boldsymbol{\tau}_{<t}\big) + \sum_{j=1}^{M} \log P_\theta\big(a_t^j \mid \mathbf{a}_t^{<j}, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) + \log P_\theta\big(r_t \mid \mathbf{a}_t, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) \Big),$$

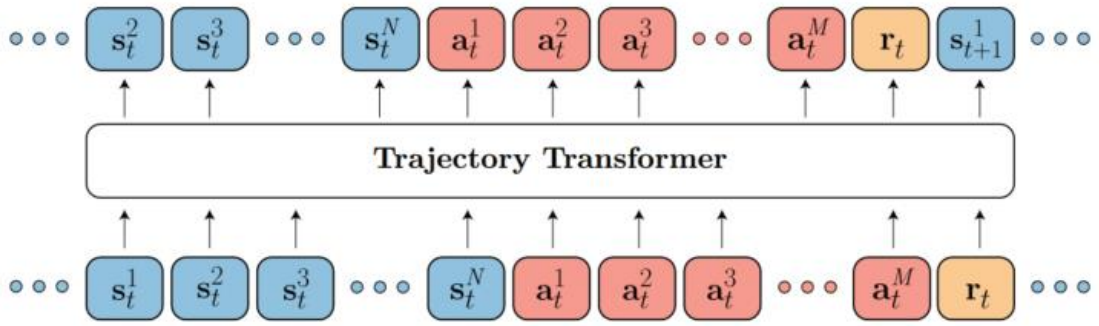


Figure 1 (Architecture) The Trajectory Transformer trains on sequences of (autoregressively discretized) states, actions, and rewards. Planning with the Trajectory Transformer mirrors the sampling procedure used to generate sequences from a language model.

[1] Offline reinforcement learning as one big sequence modeling problem. NeurIPS, 2021.

Applied LDM Case: GATO

- GATO: LDM via Self-supervised Learning
- Tokenized Data Types:
  - Texts
  - Images
  - Discrete/Continuous Values
- Application Tasks:
  - Robot Control
  - Atari Games
  - Text/Image Tasks

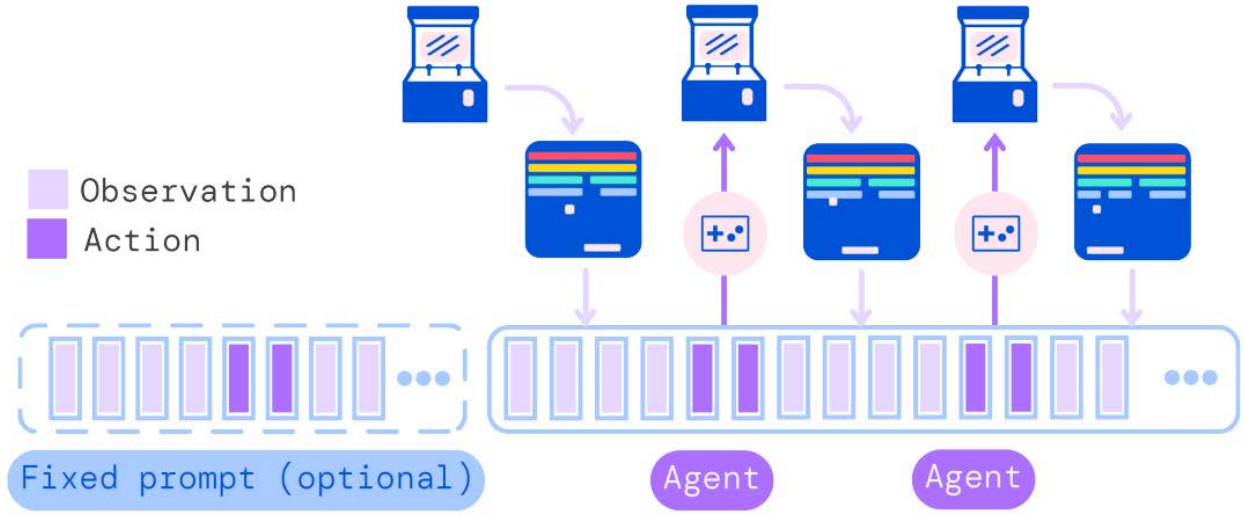


Figure 3: **Running Gato as a control policy.** Gato consumes a sequence of interleaved tokenized observations, separator tokens, and previously sampled actions to produce the next action in standard autoregressive manner. The new action is applied to the environment – a game console in this illustration, a new set of observations is obtained, and the process repeats.

[1] A Generalist Agent. DeepMind. 2022.

Advanced topics in RL

- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- Learning from demonstrations
- Large Decision Model（LDM）
- RL for LLM
- LLM and agent

RL for LLM

- RL is power in LLM alignment/finetuning
  - RLHF（PPO）: alignment with human feedback
  - GRPO: reinforcement learning finetuning for enhanced reasoning （DeepSeek-R1）
  - DAPO: more stable and open-source version of GRPO

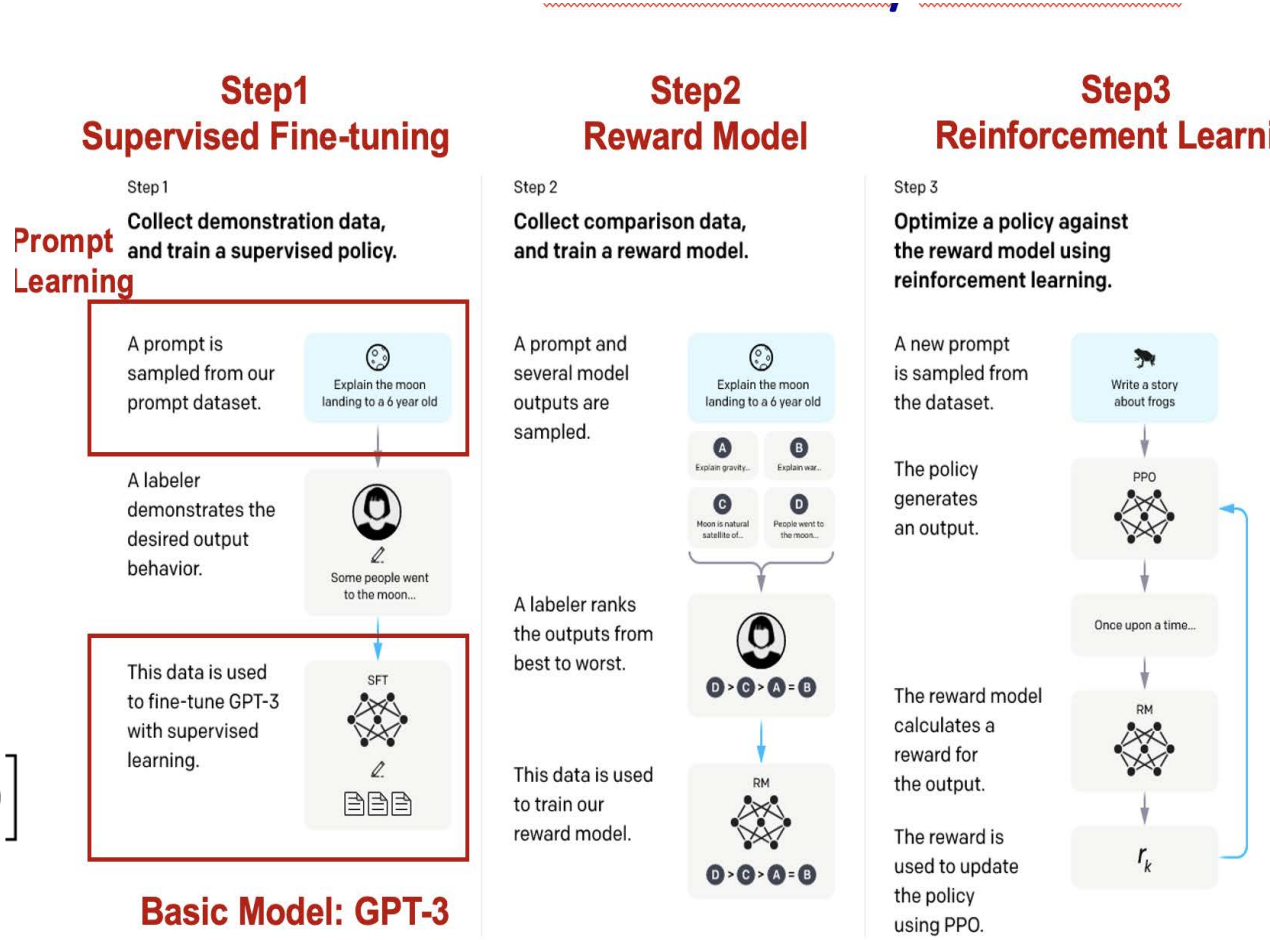
PPO and RLHF

- Actor: pretrained LLM
- Reward: a reward model by human labelled data
- Algorithm: Proximal Policy Optimization (PPO)

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Small and save update



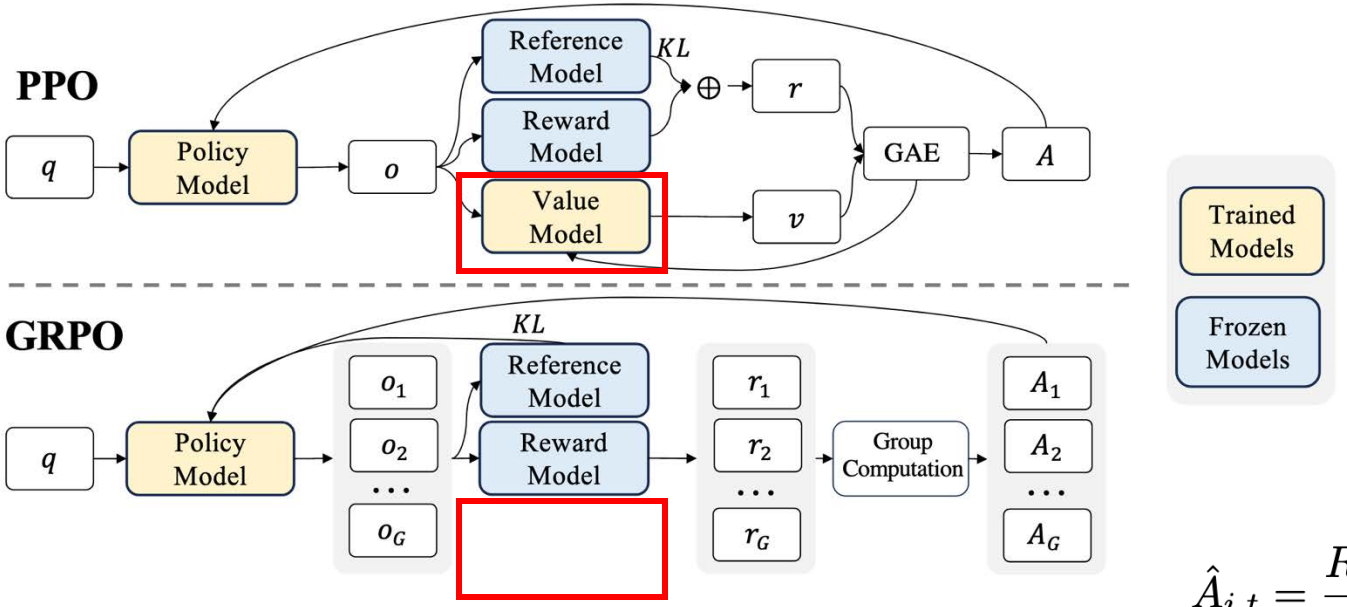
[1] Training language models to follow instructions with human feedback. NeurIPS, 2022.
[2] Proximal Policy Optimization Algorithms. 2017.

GRPO for LLM reasoning

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \qquad \delta_l = R_l + \gamma V(s_{l+1}) - V(s_l)$$

Advantage estimation: Auxiliary value model
(Similar size as the LLM itself)

$$\mathcal{J}_{\mathrm{PPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},o_{\le t}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q)}\left[\min\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\mathrm{old}}}(o_t \mid q,o_{<t})}\hat{A}_t,\ \mathrm{clip}\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\mathrm{old}}}(o_t \mid q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right]$$



$$\hat{A}_{i,t} = \frac{R_i - \mathrm{mean}(\{R_i\}_{i=1}^G)}{\mathrm{std}(\{R_i\}_{i=1}^G)}$$

Advantage estimation: Monte Carlo rollout and group computation

- Substantial parameter reduction
- Effective for LLM reasoning (DeepSeek-R1)

[1] DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024.

DAPO: improving the stability of GRPO

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}$$

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}$$

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) \right) \right]$$

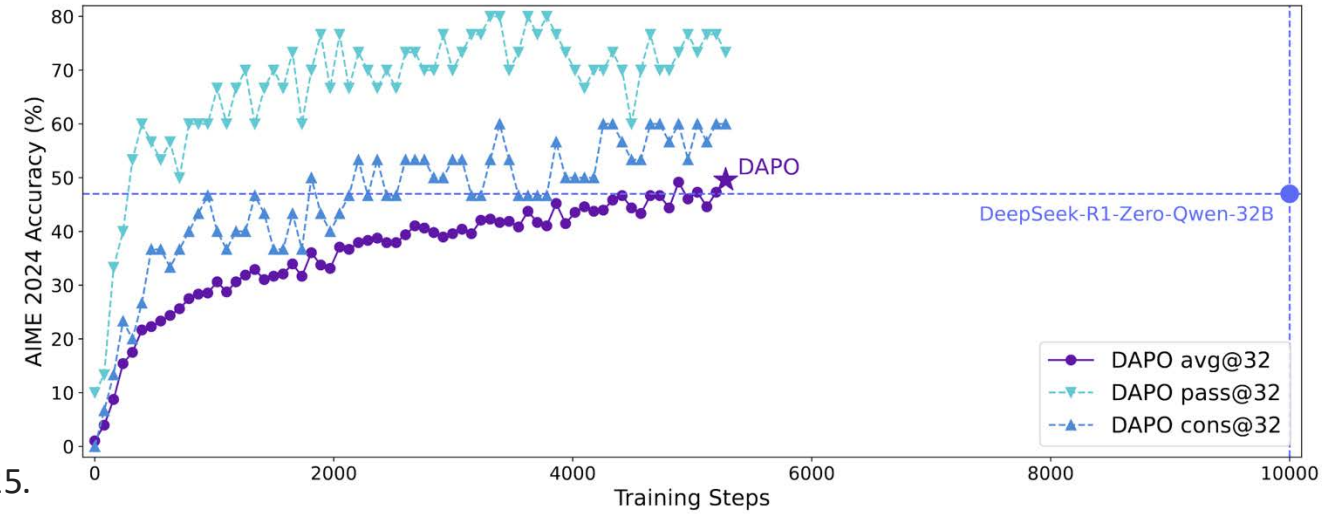
Token-level loss: Averaging tokens in all chains equally

Clip-Higher: Better exploration for low-probability tokens

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right) \right]$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G,$$

Dynamic Sampling: Exclude all-right and all-wrong sample groups

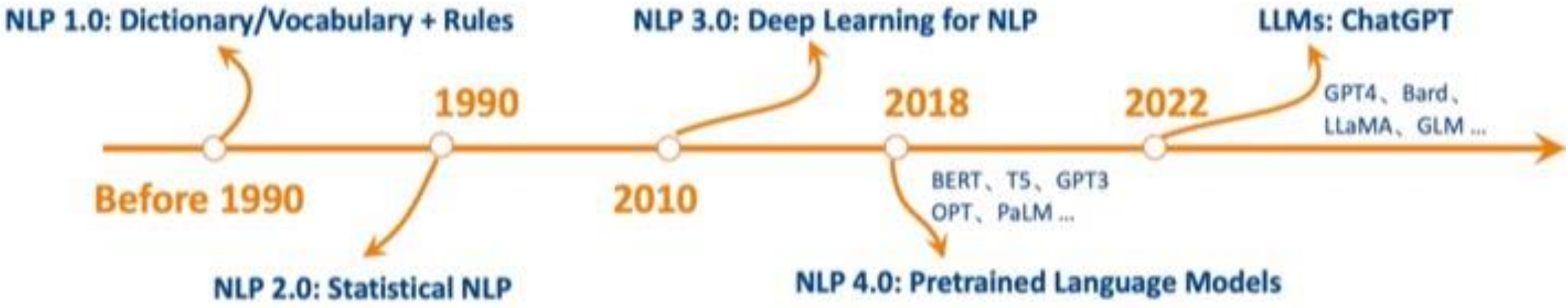


[1] DAPO: An Open-Source LLM Reinforcement Learning System at Scale. 2025.

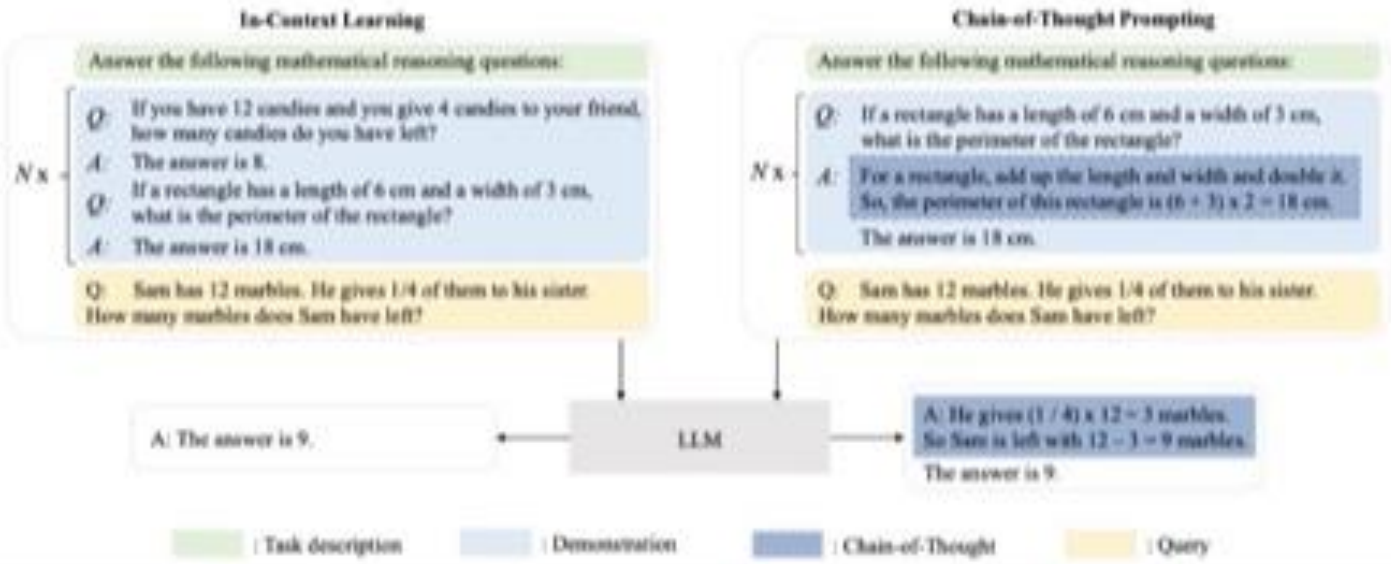
Advanced topics in RL

- Improvements on DQN
- Bridging the gap between model-free and model-based RL
- Offline RL
- Learning from demonstrations
- Large decision model（LDM）
- RL for LLM
- LLM and agent

LLM: development and capabilities



- Emergent capabilities of LLM
  - Rich knowledge & Language Capabilities
  - Instruction following
  - In-context learning
  - Chain-of-thought
  - Planning
  - ...

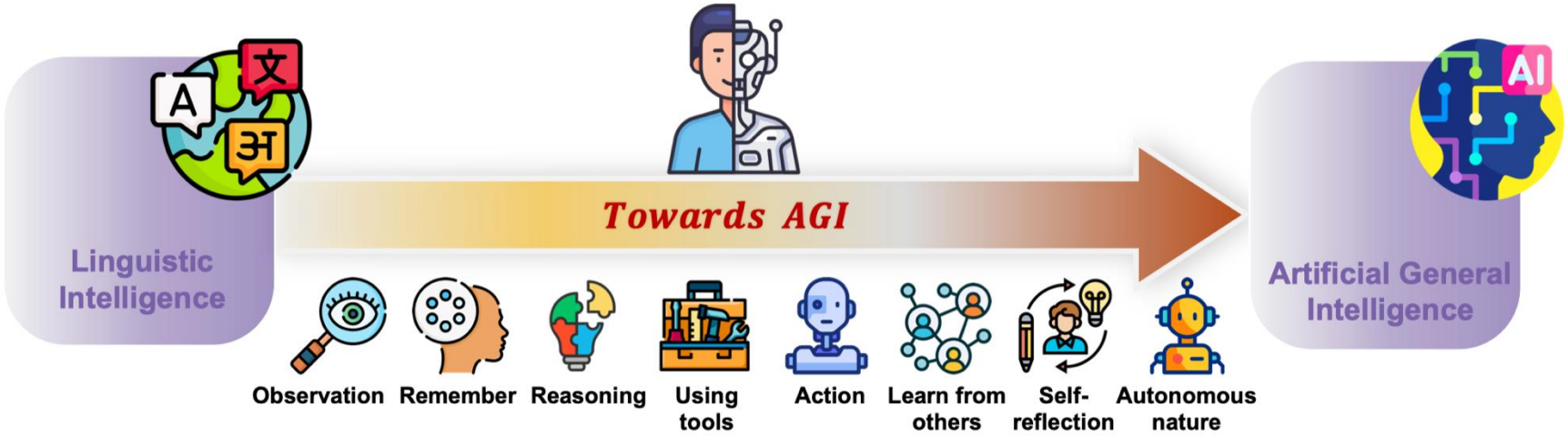


From LLM to LLM Agents

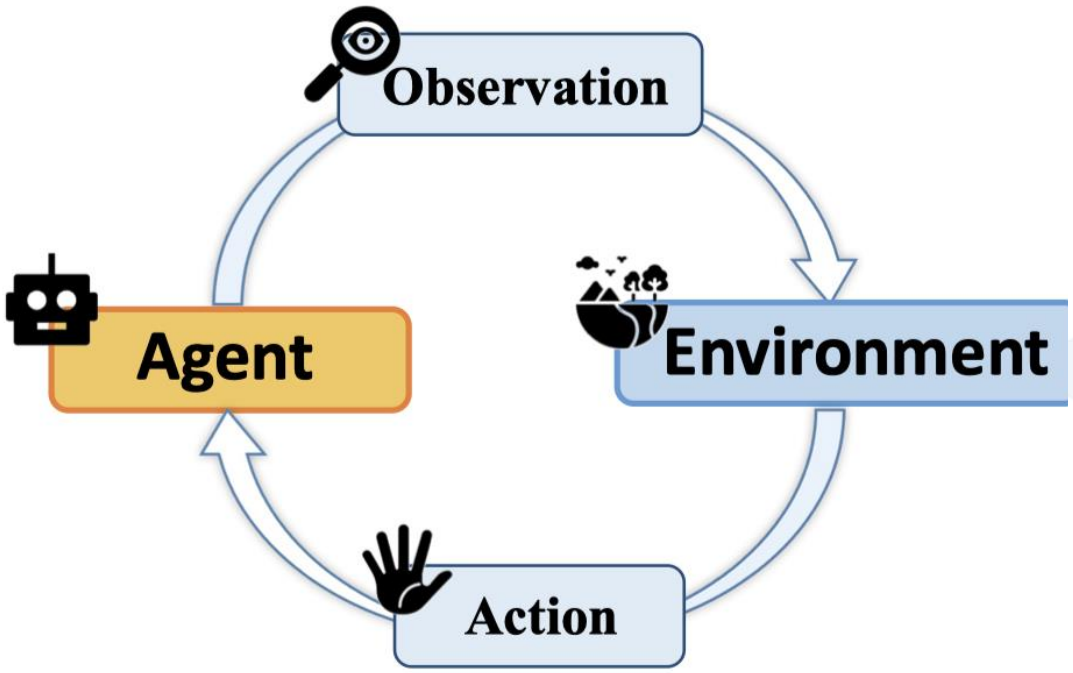
- LLMs have many strengths
  - They have mastered the rules of **human language**, which is one of the most complicated tasks
  - They have rich knowledge, e.g., **common sense knowledge** & domain knowledge..
  - They have good **reasoning ability**, such as the chain of thoughts reasoning
  - And many more ...

From LLM to LLM Agents

- LLMs have many weaknesses
  - They often make mistakes (corpuses for training are not perfect)
  - Hallucination
  - May be not so good at logical reasoning/fine-grained visual understanding/etc.
- Large LLMs exhibit characteristics of artificial general intelligence (AGI), which has cognitive abilities similar to that of human beings.
- But, however, LLMs are not AGI



From LLM to LLM Agents



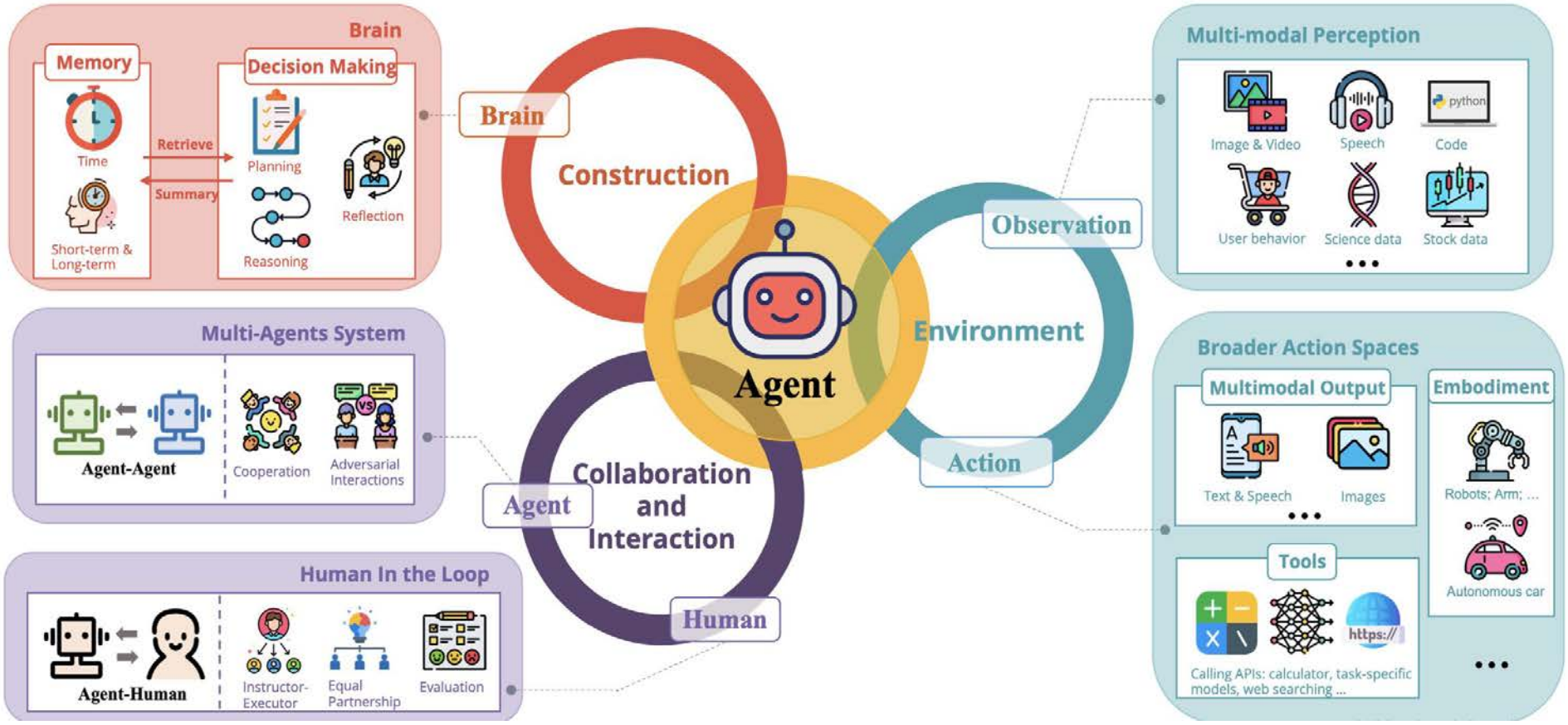
Environment

- The external **context** or **surroundings** in which the agent operates and makes decisions.

- Human & Agents' behaviors
- External database and knowledges

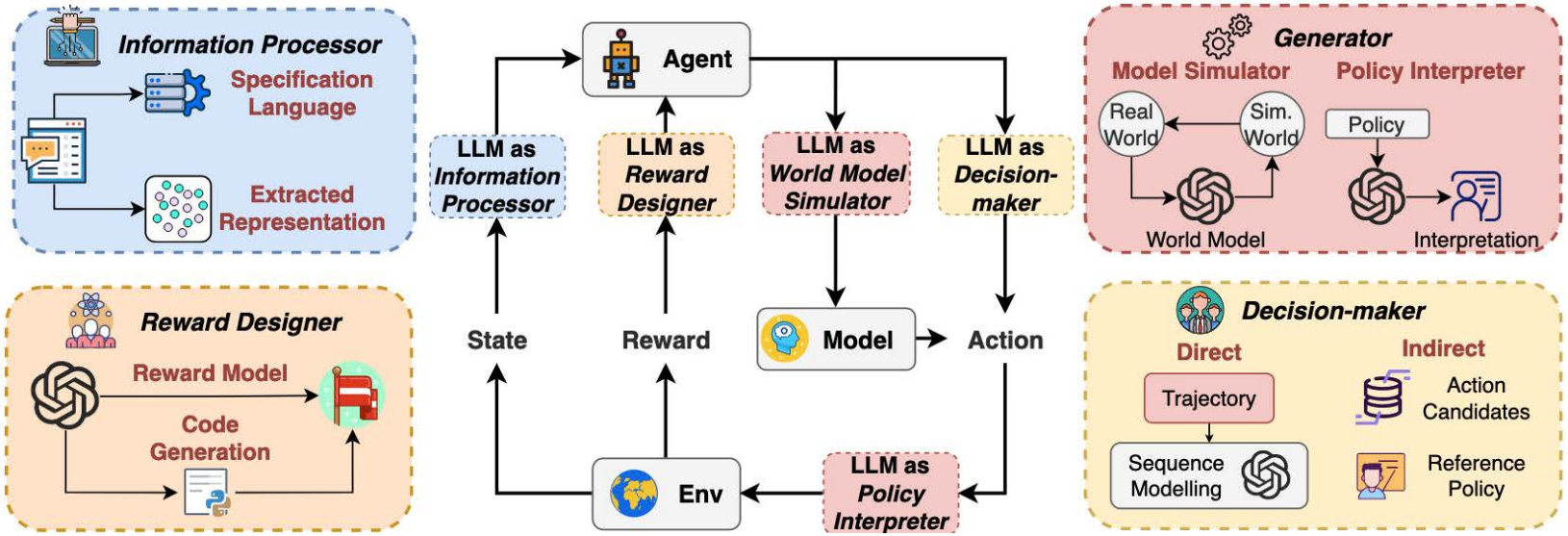
- Virtual & Physical environment

From LLM to LLM Agents



Agent: Construction, Environment, and Interaction

LLM and Agent vs RL

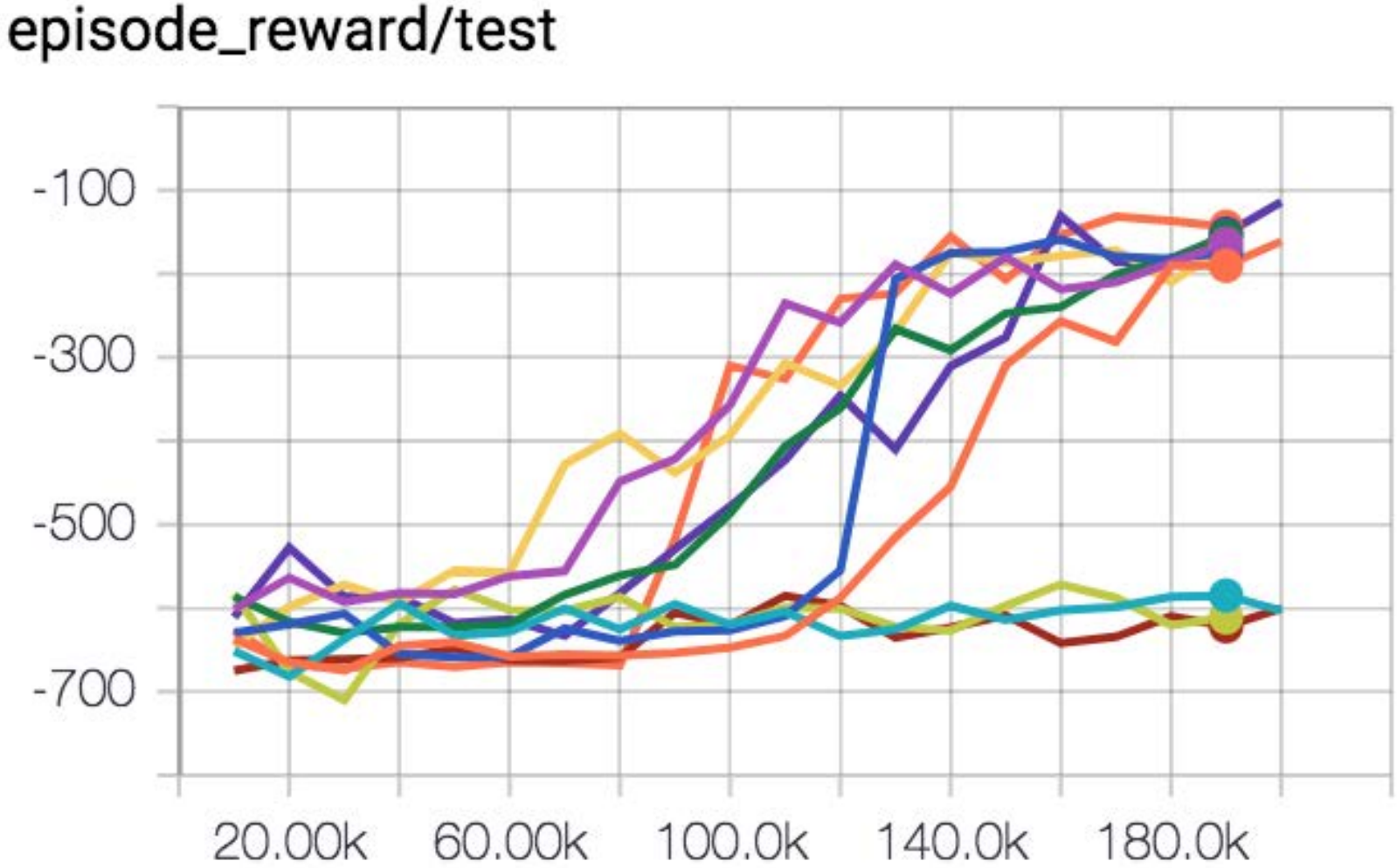


- LLM/Agent help RL
  - Design action, reward, etc.

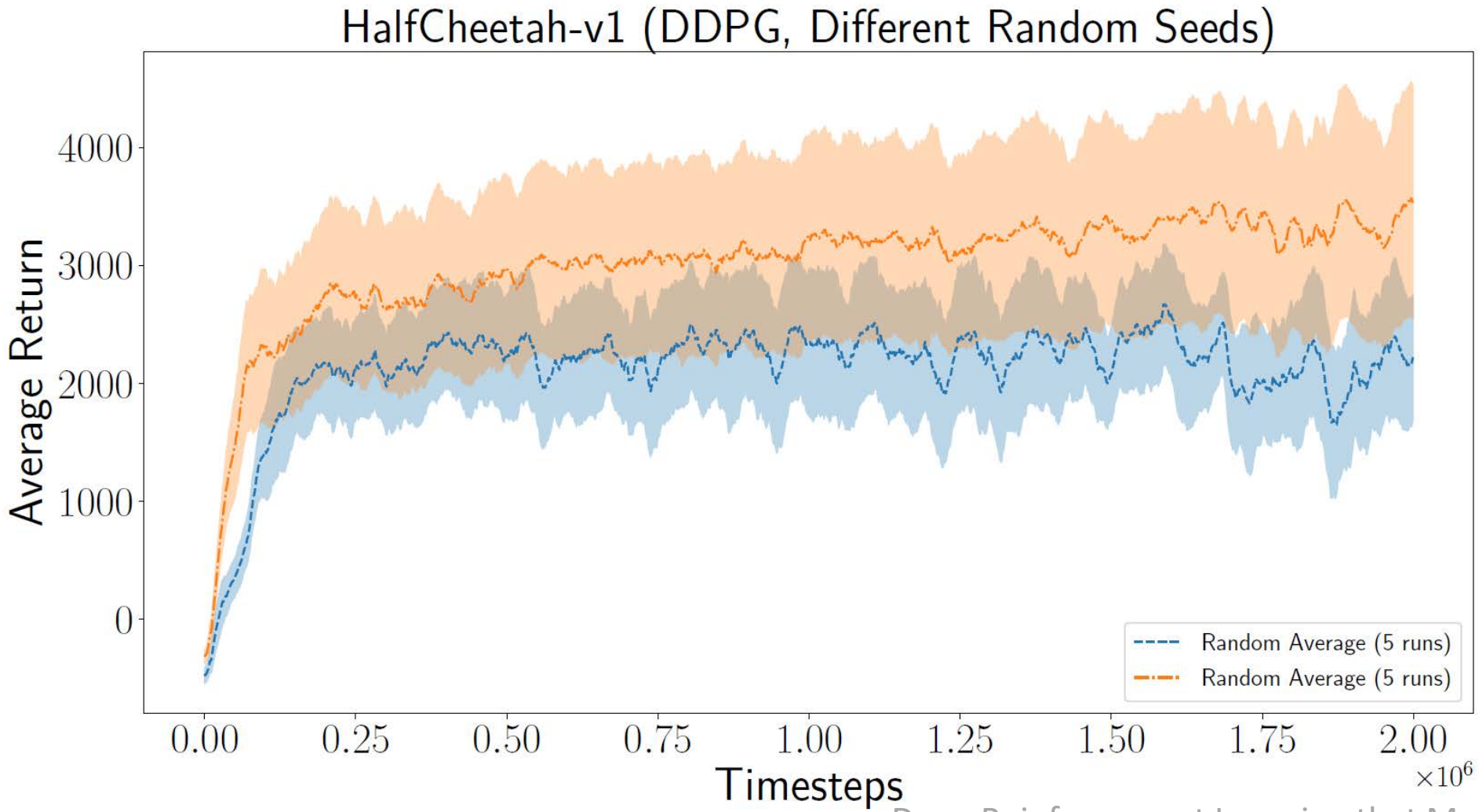
    any part of RL
- LLM/Agent replace RL
  - Lots of open problem

Challenges and Opportunities

1. Robustness – random seeds

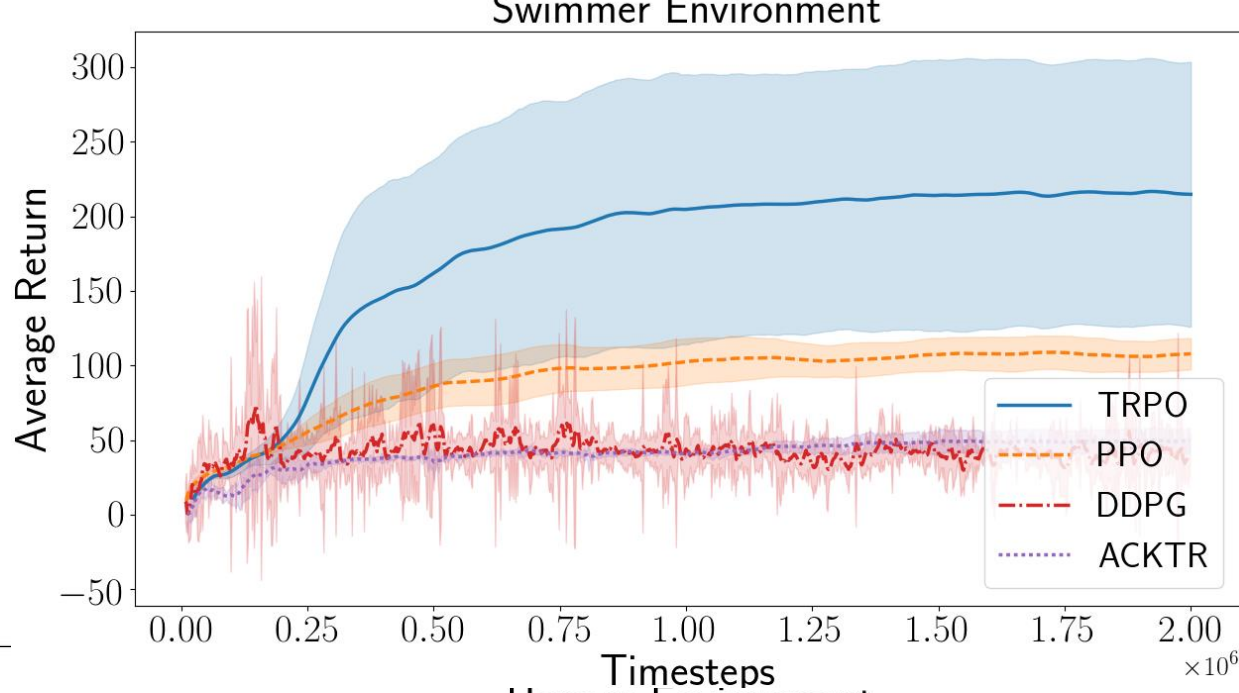


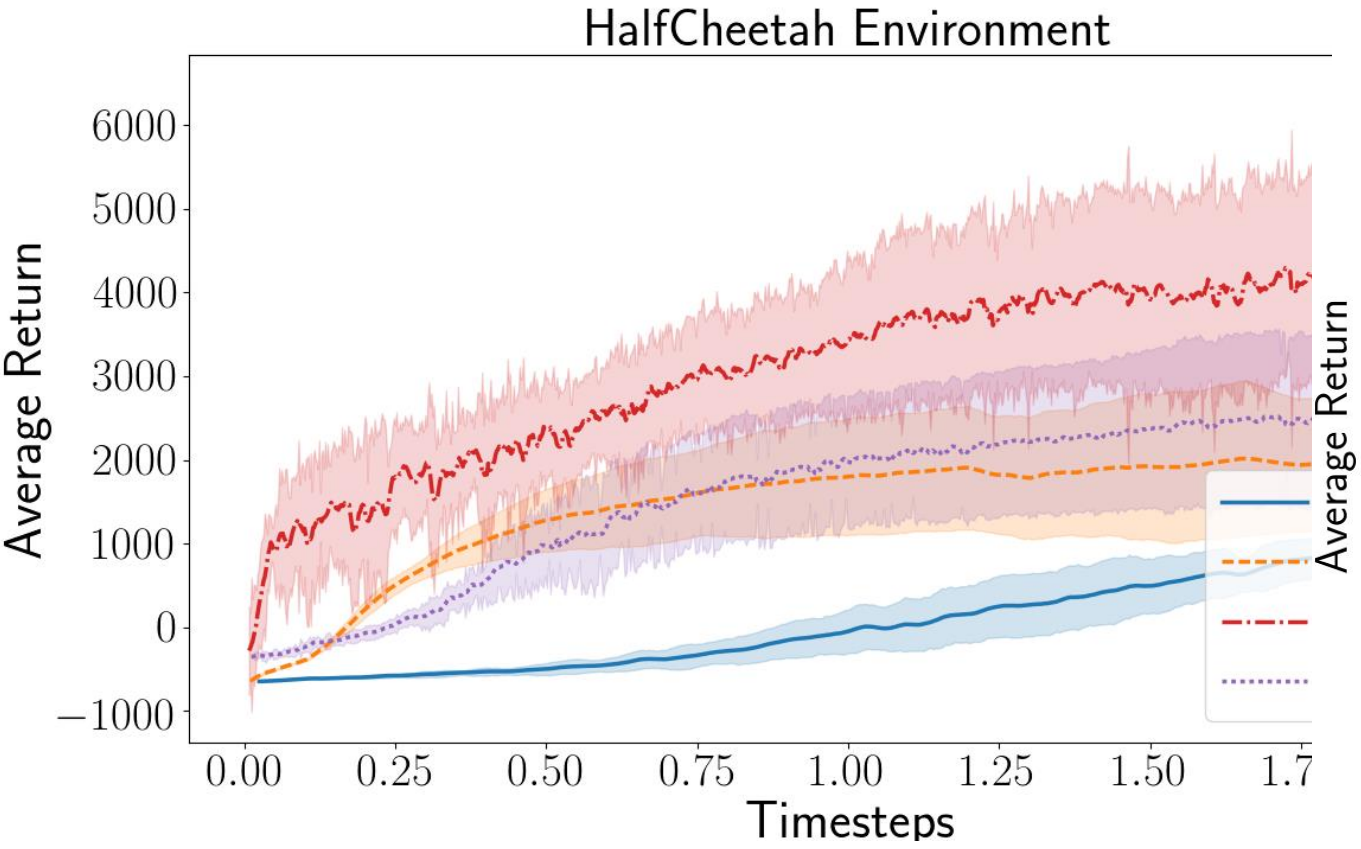
1. Robustness – random seeds

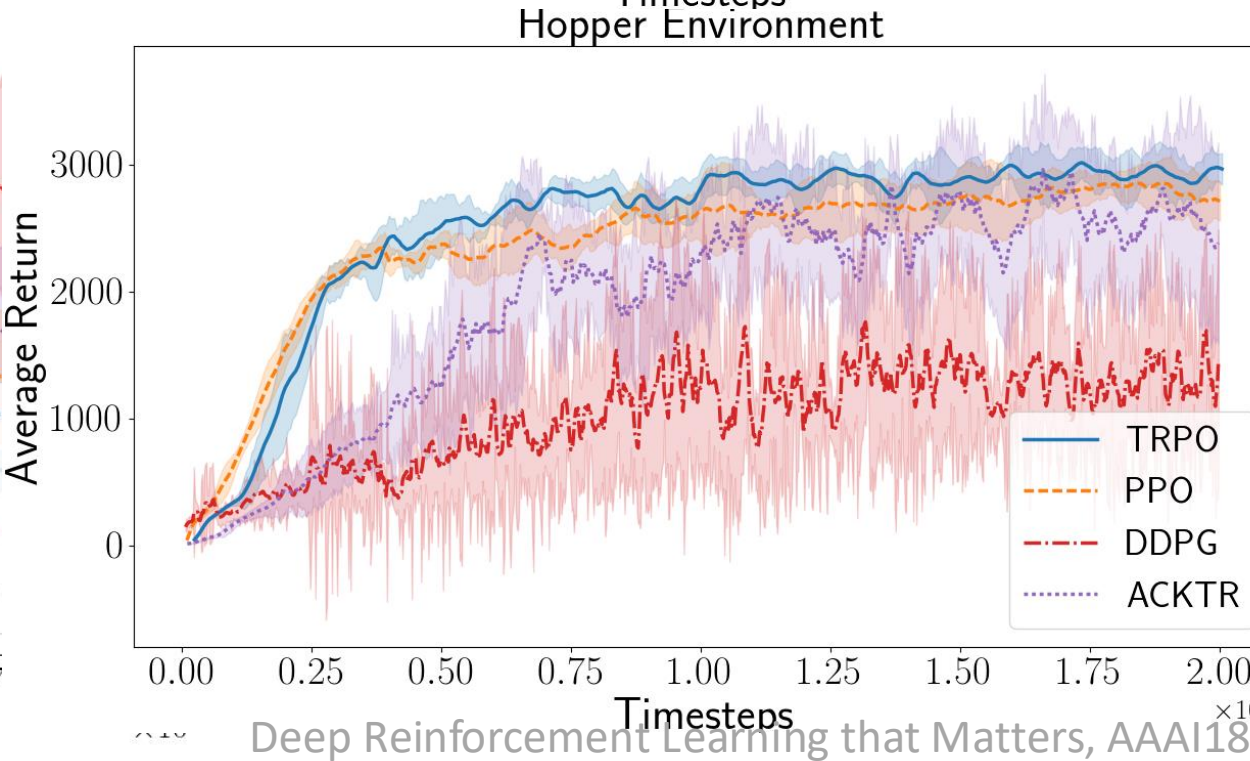


Deep Reinforcement Learning that Matters, AAAI18

2. Robustness – across task







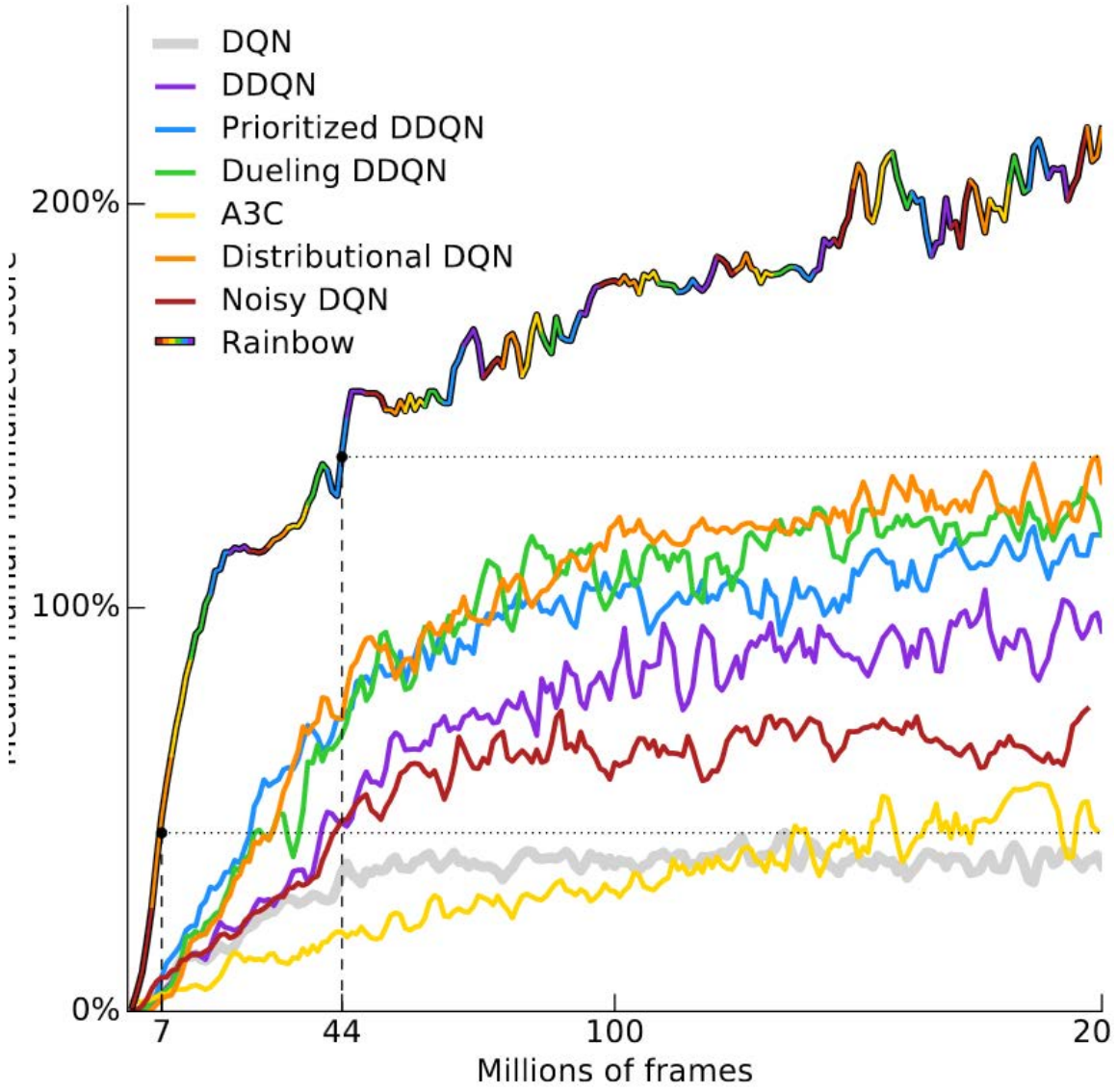
Deep Reinforcement Learning that Matters, AAAI18

As a Comparison

- ResNet performs pretty well on various kinds of tasks
  - Object detection
  - Image segmentation
  - Go playing
  - Image generation
  - …

3. Learning - sample efficiency

- Supervised learning
  - Learning from oracle
- Reinforcement learning
  - Learning from trial and error

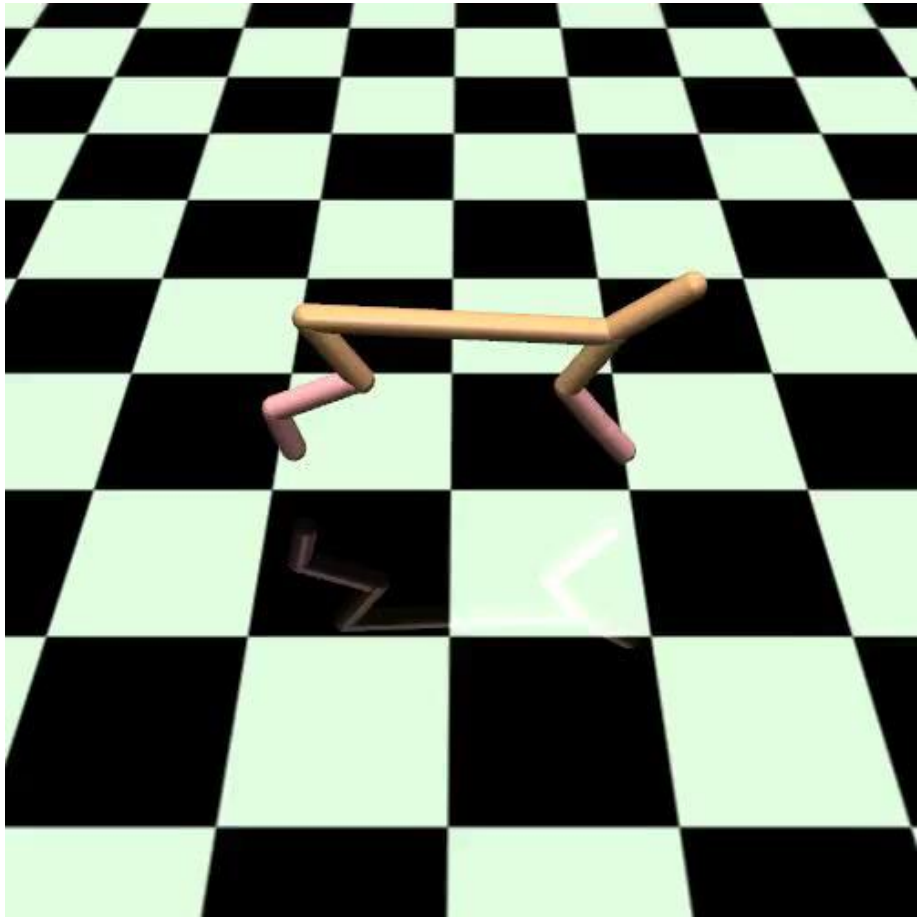


Rainbow: Combining Improvements in Deep Reinforcement Learning

Multi-task/transfer learning

- Humans can’t learn individual complex tasks from scratch.
- Maybe our agents shouldn’t either.
- We ultimately want our agents to learn many tasks in many environments
  - learn to learn new tasks quickly (Duan et al. ’17, Wang et al. ’17, Finn et al. ICML ’17)
  - share information across tasks in other ways (Rusu et al. NIPS ’16, Andrychowicz et al. ‘17, Cabi et al. ’17, Teh et al. ’17)
- Better exploration strategies

4. Optimization – local optima



5. No/sparse reward

Real world interaction:

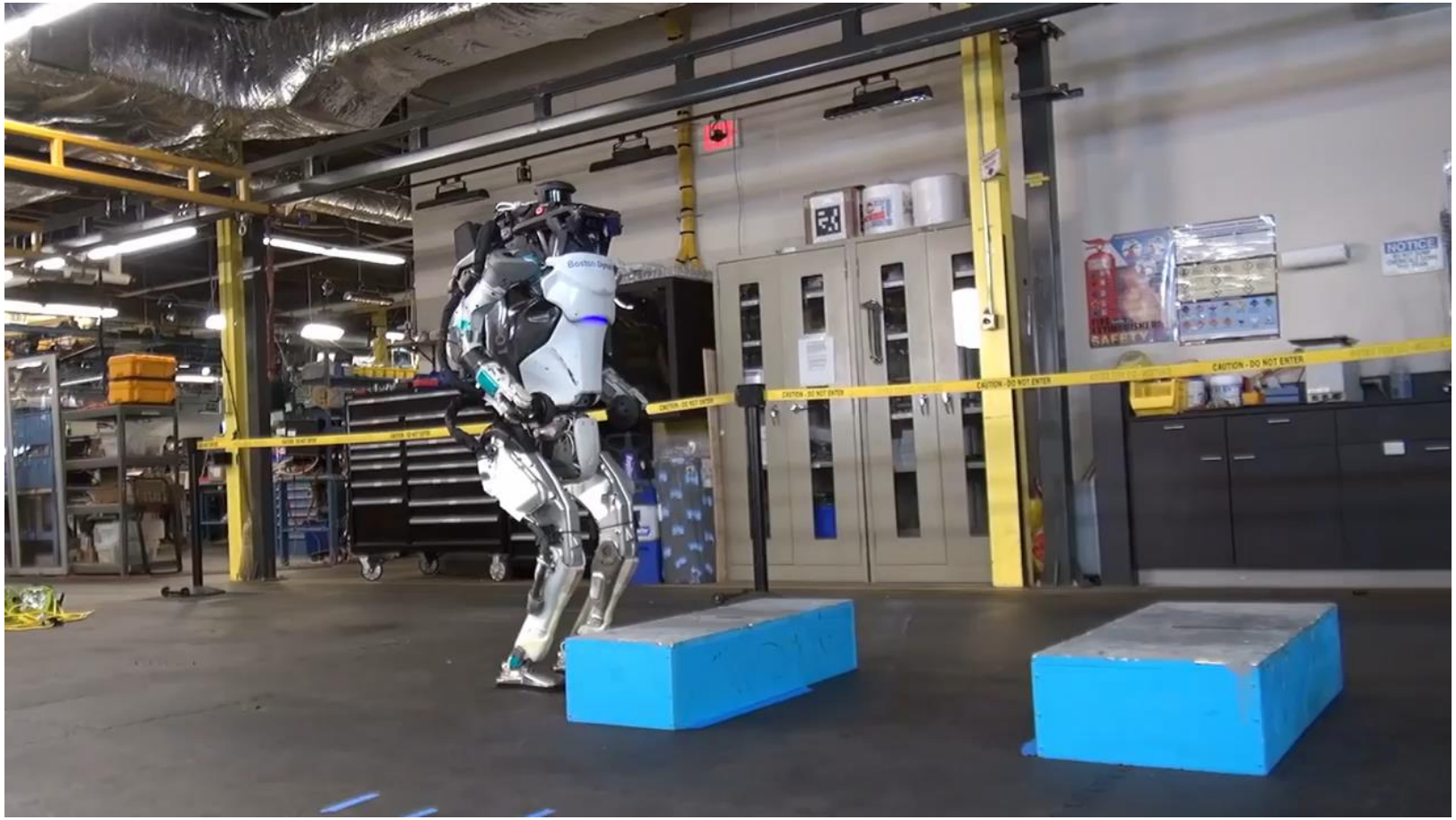
- Usually no (visible) immediate reward for each action
- Maybe no (visible) explicit final reward for a sequence of actions
- Don’t know how to terminate a sequence

Consequences:

- Most DRL algos are for games or robotics
  - Reward information is defined by video games in Atari and Go
  - Within controlled environments

- Scalar reward is an extremely sparse signal, while at the same time, humans can learn without any external rewards.
  - Self-supervision (Osband et al. NIPS ’16, Houthooft et al. NIPS ’16, Pathak et al. ICML ’17, Fu*, Co-Reyes* et al. ‘17, Tang et al. ICLR ’17, Plappert et al. ‘17)
  - options & hierarchy (Kulkarni et al. NIPS ’16, Vezhnevets et al. NIPS ’16, Bacon et al. AAAI ’16, Heess et al. ‘17, Vezhnevets et al. ICML ’17, Tessler et al. AAAI ’17)
  - leveraging stochastic policies for better exploration (Florensa et al. ICLR ’17, Haarnoja et al. ICML ’17)
  - auxiliary objectives (Jaderberg et al. ’17, Shelhamer et al. ’17, Mirowski et al. ICLR ’17)

6. Is DRL a good choice for a task?



7. Imperfect-information games and multi-agent games

- No-limit heads up Texas Hold'Em
  - Libratus (Brown et al, NIPS 2017)
  - DeepStack (Moravčík et al, 2017)

Refer to Prof. Bo An's talk

Opportunities

- Improve robustness (e.g., w.r.t random seeds and across tasks)
- Improve learning efficiency
- Better optimization
- Define reward in practical applications
- Identify appropriate tasks
- Imperfect information and multi-agent games

Applications

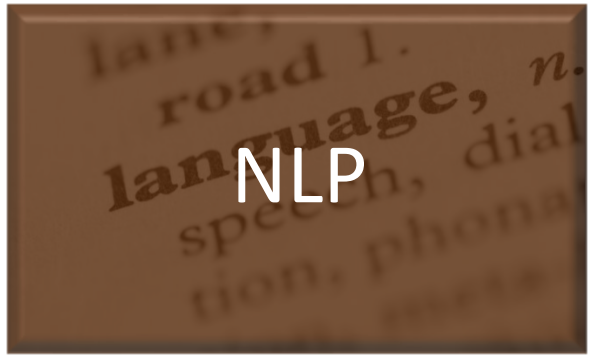

AlphaGo
Game

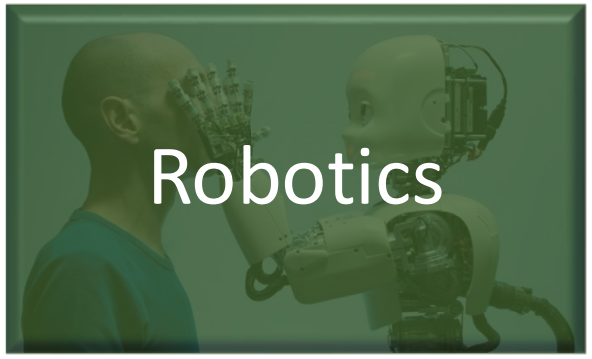

Neuro Science


Music & Movie


Healthcare


Trading

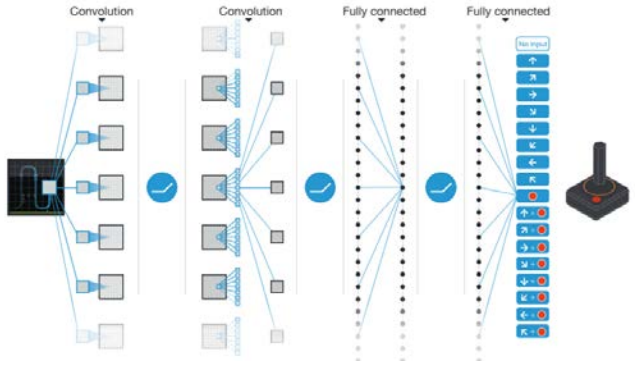

NLP

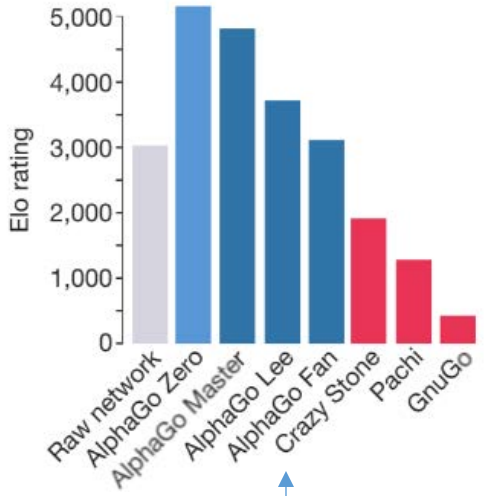

Robotics


Education


LLM Reasoning

Game





Atari games

AlphaGo Zero

OpenAI Five

… 1980 2013 2016 2017 2018 2019 2020 …

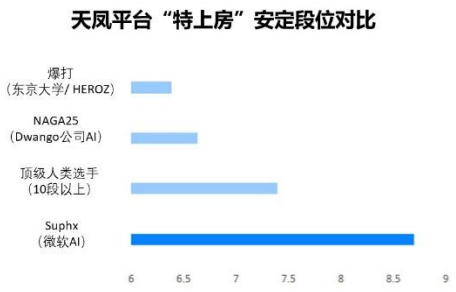
1. trial and error
2. optimal control

AlphaGo

AlphaGo

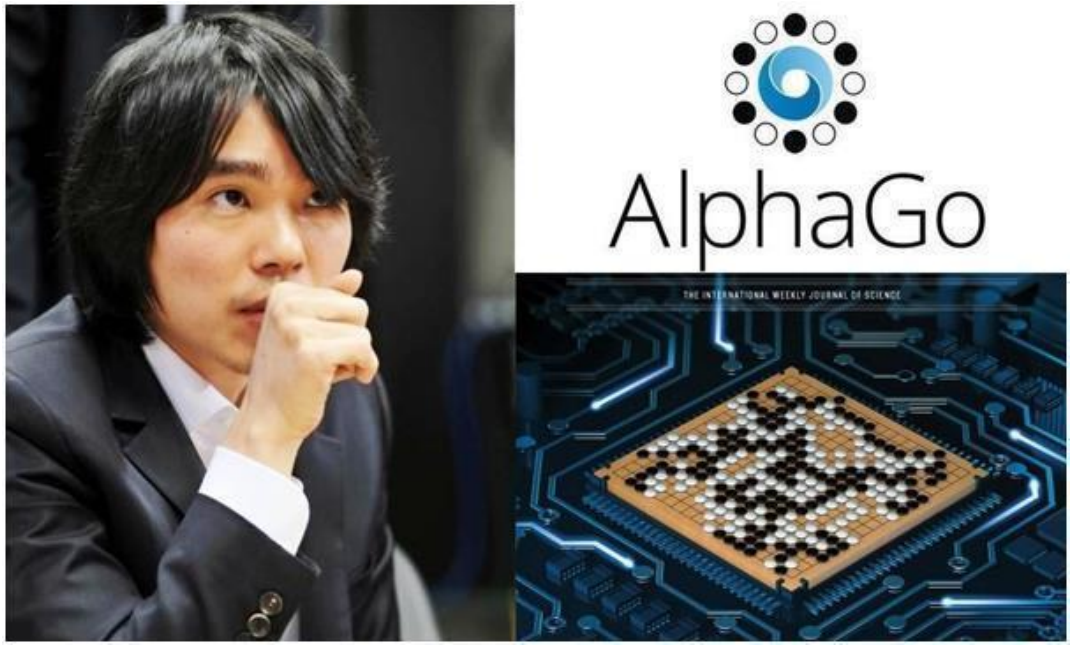
AlphaStar

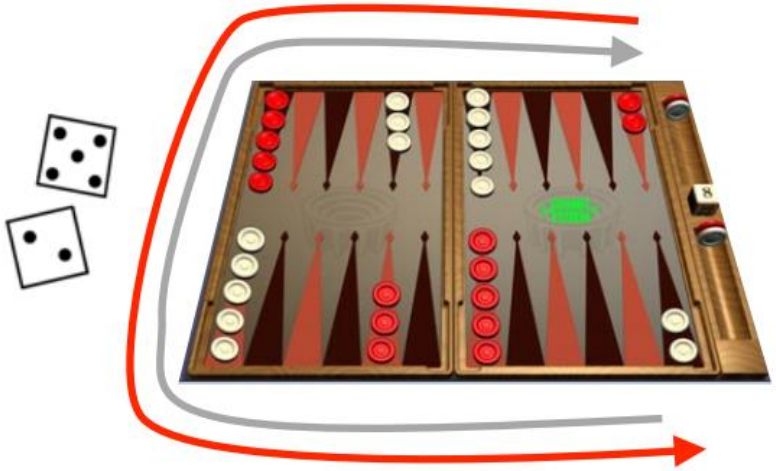
Suphx



Game

- RL for Game
  - Sequential Decision Making
  - Delayed Reward





TD-Gammon

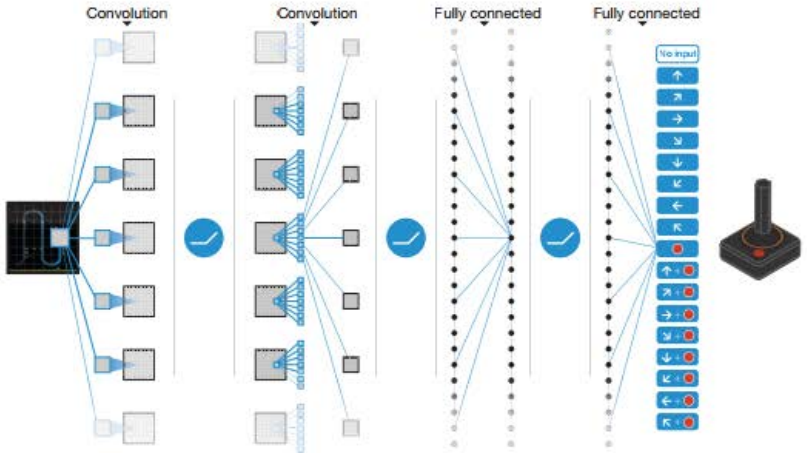
Atari Games

Game



- Atari Games
  - Learned to play 49 games for the Atari 2600 game console, without labels or human input, from self-play and the score alone
  - Learned to play better than all previous algorithms and at human level for more than half the games

mapping raw screen pixels

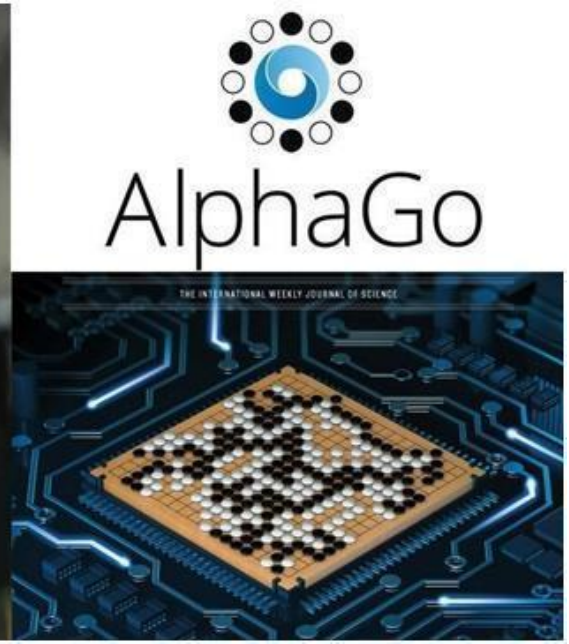


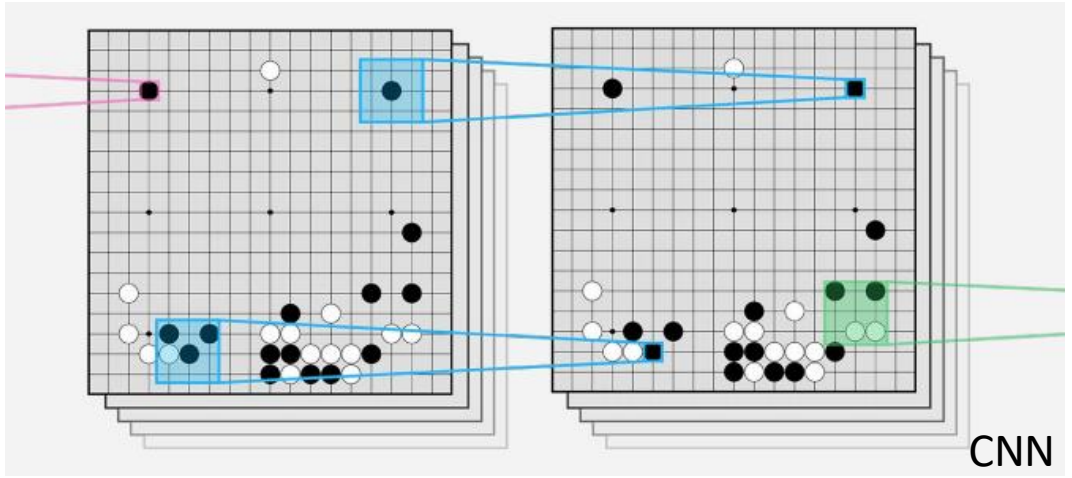
to predictions of final score for each of 18 joystick actions

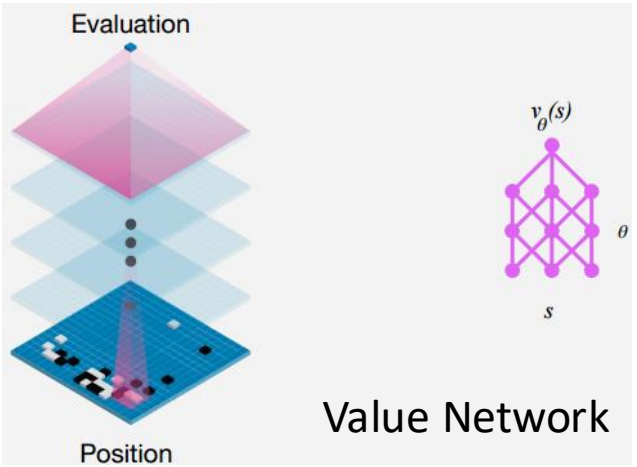
Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

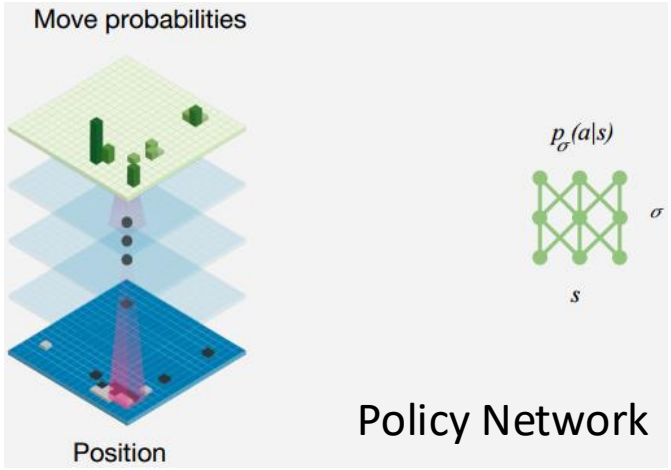
Game

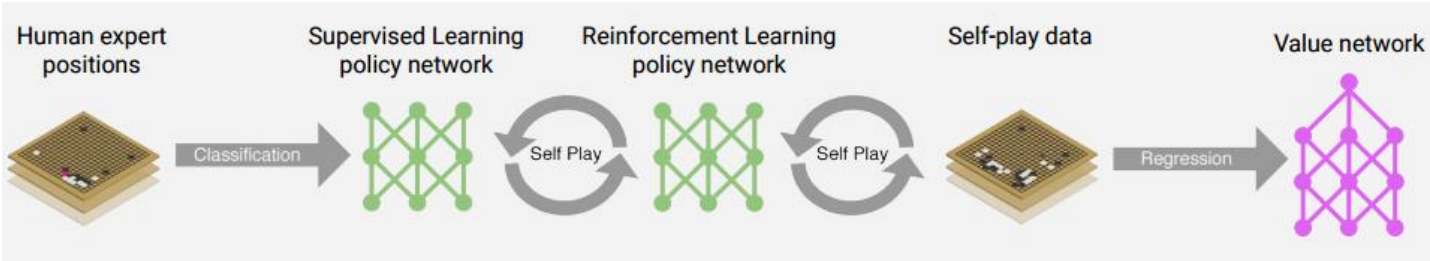
- AlphaGo 4-1
- Master(AlphaGo++) 60-0











http://icml.cc/2016/tutorials/AlphaGo-tutorial-slides.pdf

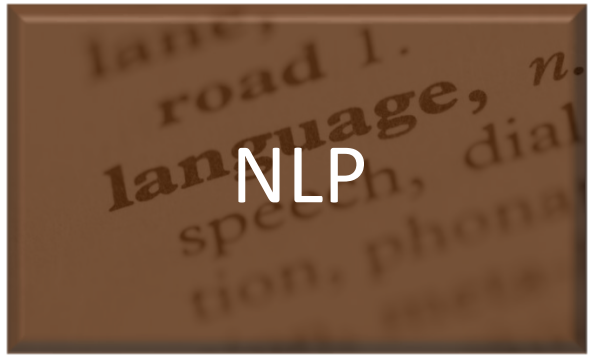

AlphaGo
Game

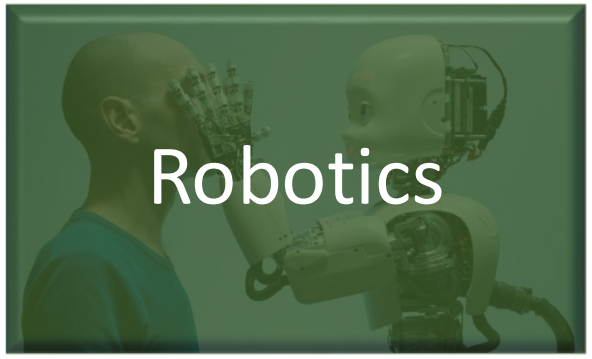

Neuro Science


Music & Movie


Healthcare


Trading


NLP


Robotics


Education


LLM Reasoning

Neuro Science

Microsoft

siam Fellow

The world presents animals/humans with a huge reinforcement learning problem (or many such small problems)

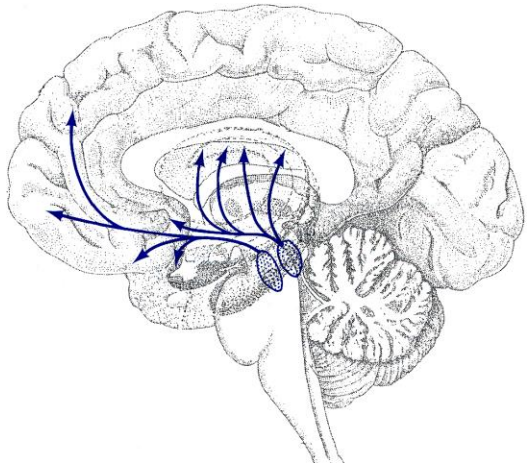
Neuro Science

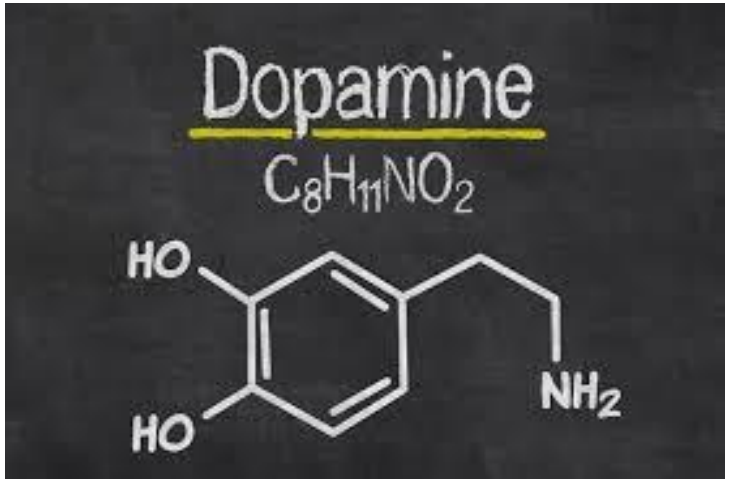
- How can the brain realize these? Can RL help us understand the brain's computations?

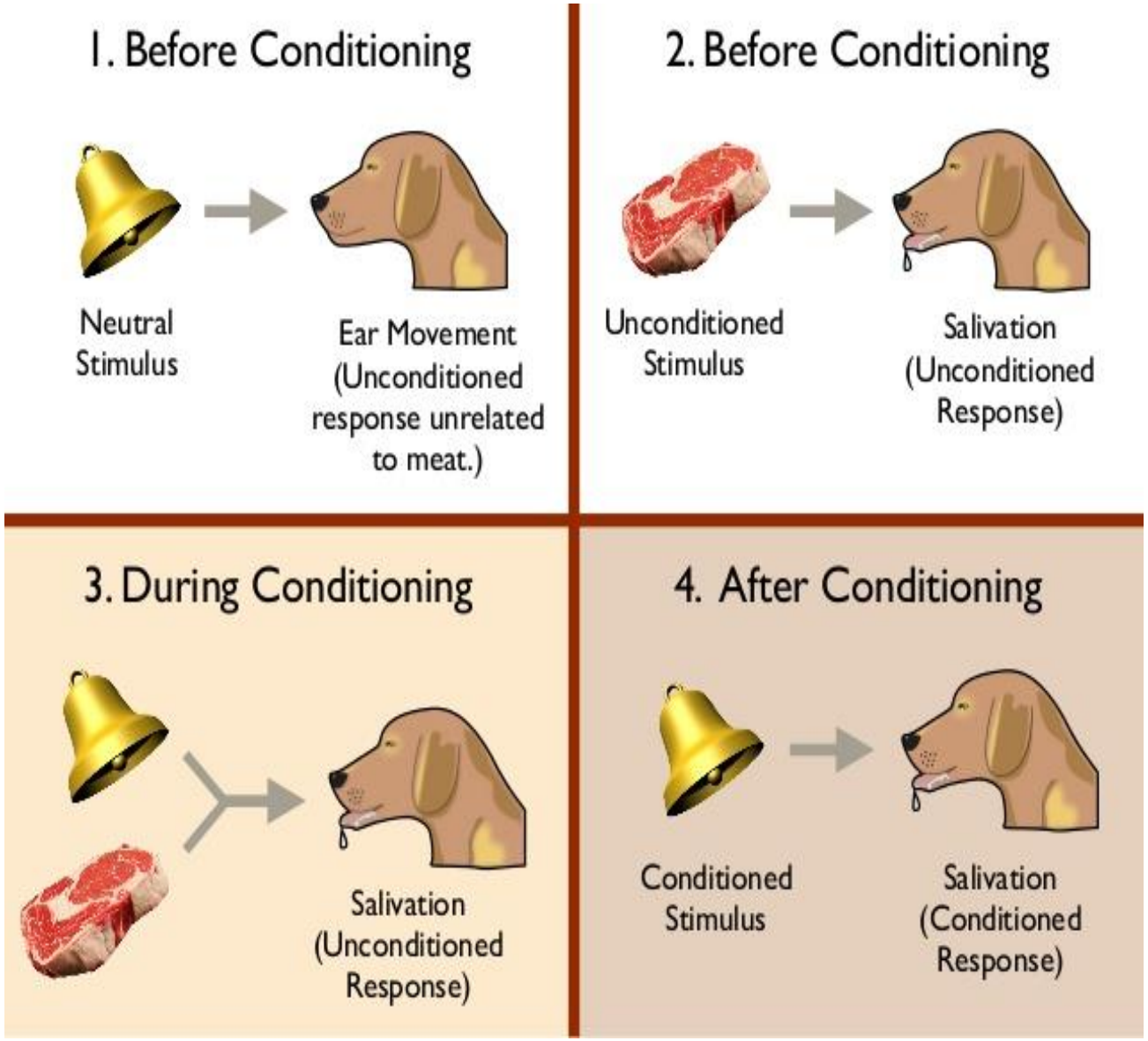
- Reinforcement learning has revolutionized our understanding of learning in the brain in the last 20 years.
  - A success story: Dopamine and prediction errors

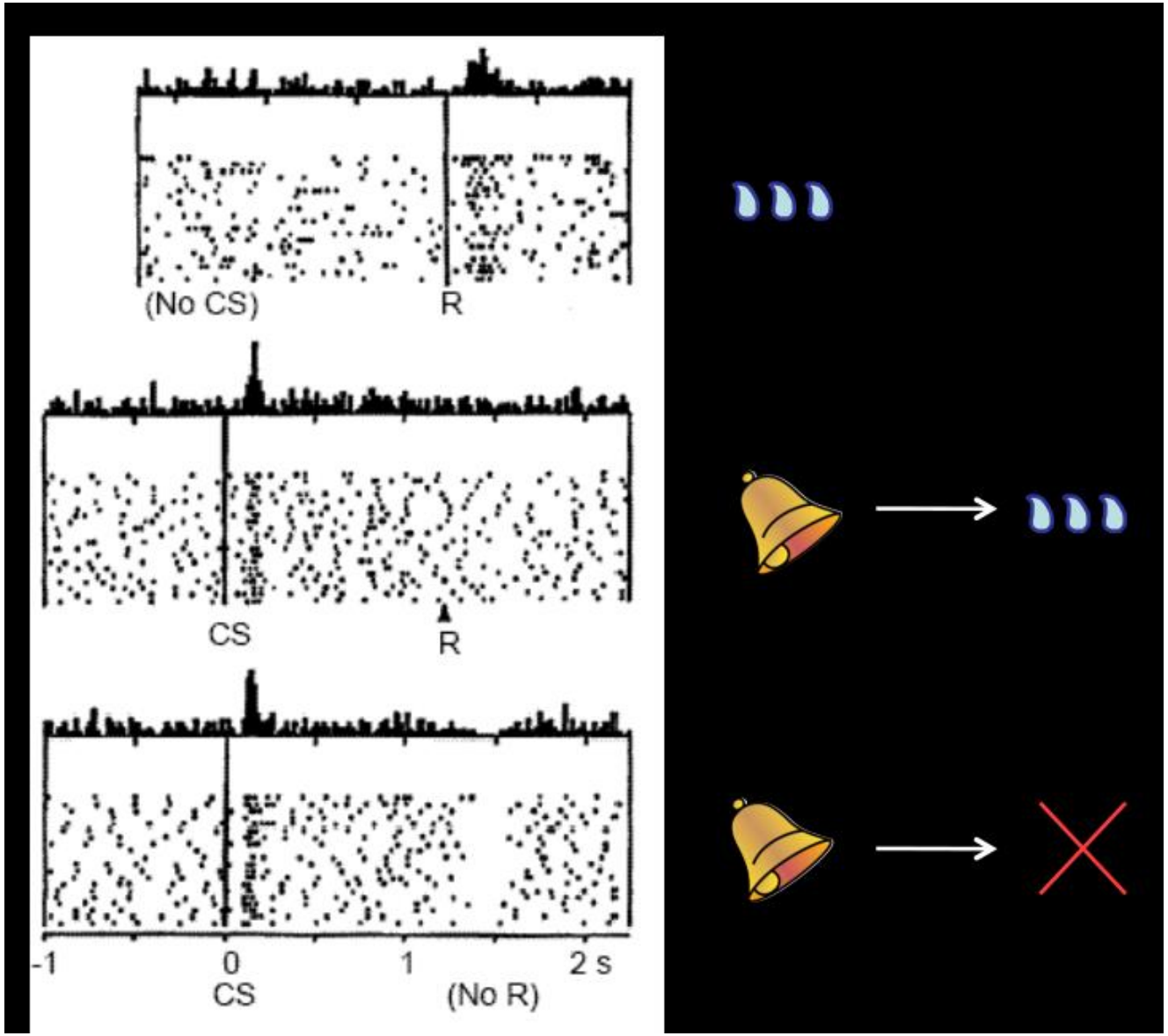
Yael Niv. The Neuroscience of Reinforcement Learning. Princeton University. ICML'09 Tutorial

What is dopamine?



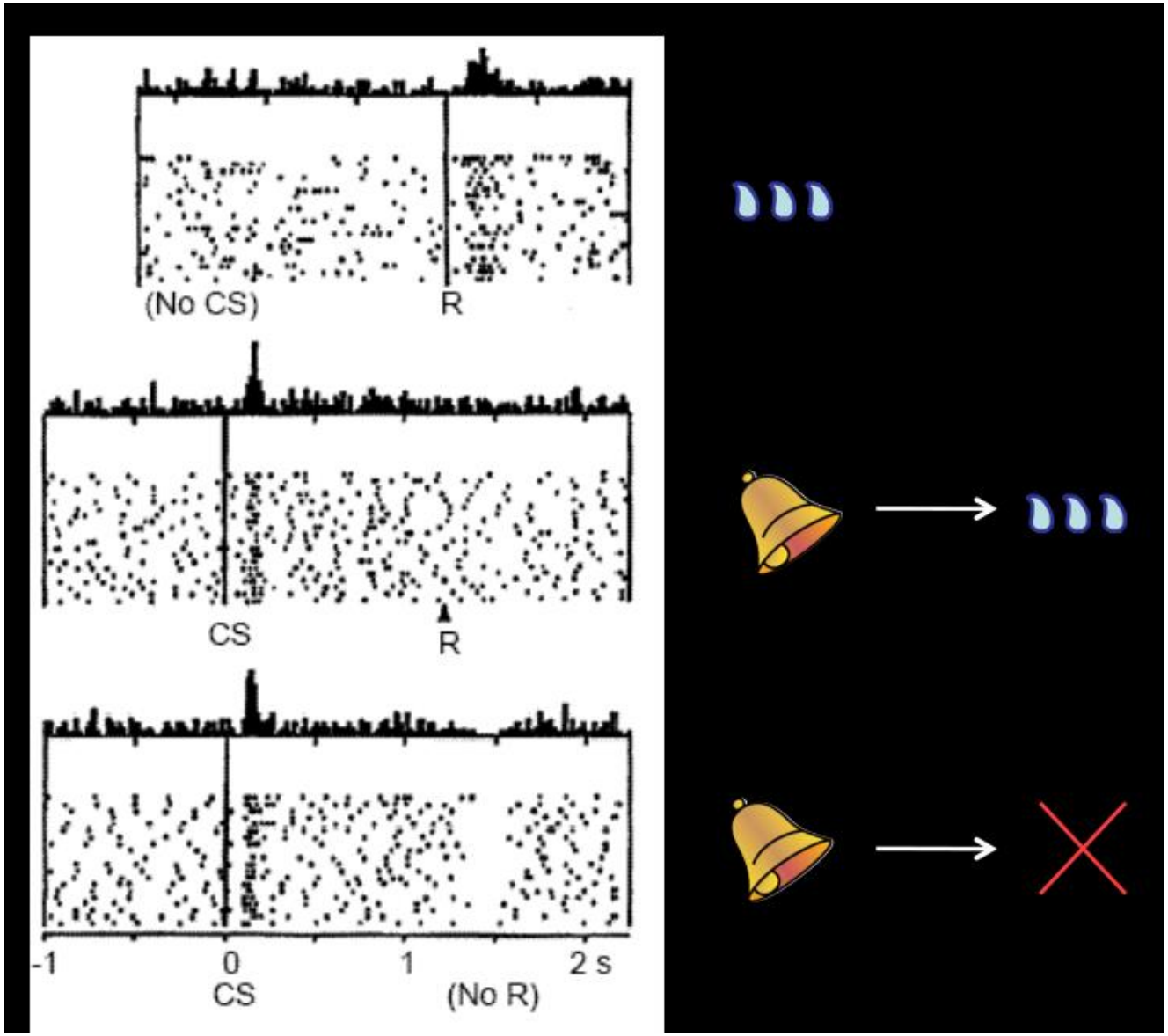



1. Before Conditioning
Neutral Stimulus
Ear Movement (Unconditioned response unrelated to meat.)
2. Before Conditioning
Unconditioned Stimulus
Salivation (Unconditioned Response)
3. During Conditioning
Salivation (Unconditioned Response)
4. After Conditioning
Conditioned Stimulus
Salivation (Conditioned Response)


(No CS)
R
CS
R
-1
0
CS
1
(No R)
2 s

The idea: Dopamine encodes a temporal difference reward prediction error

(Montague, Dayan, Barto mid 90's)



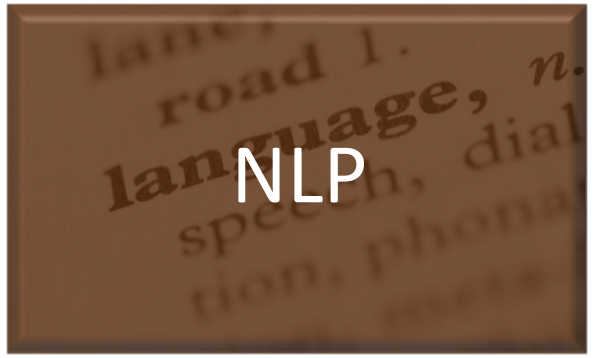

AlphaGo
Game

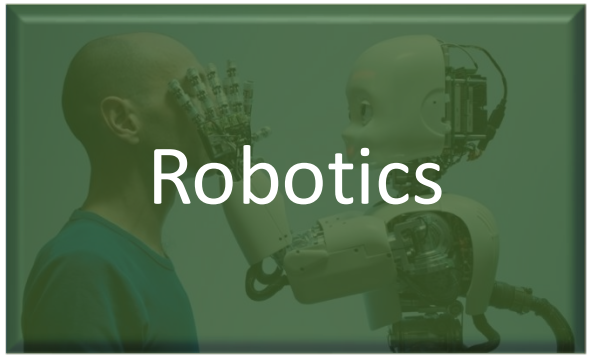

Neuro Science


Music & Movie


Healthcare


Trading


NLP

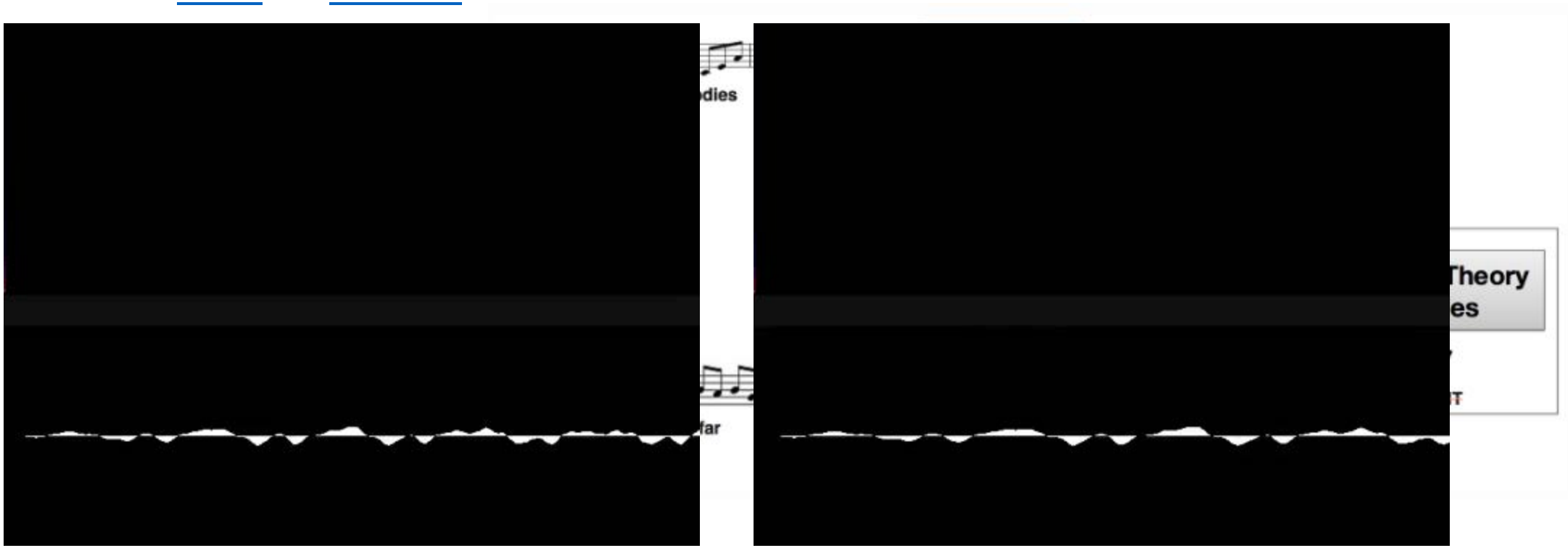

Robotics


Education


LLM Reasoning

Music & Movie

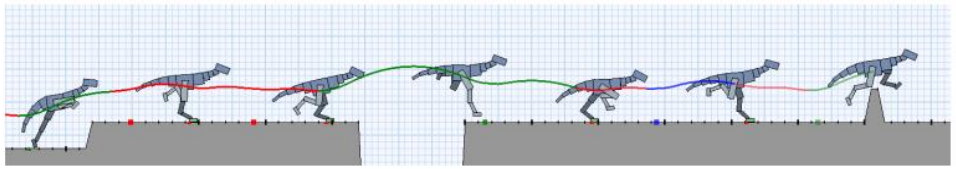
- Music
  - Tuning Recurrent Neural Networks with Reinforcement Learning
    - LSTM v.s. RL tuner



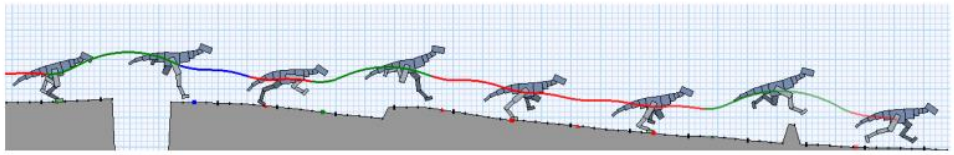
https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning/

Music & Movie

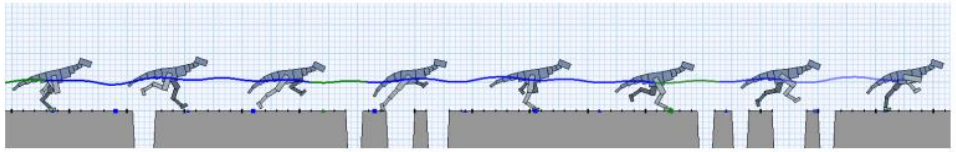
- Movie
  - Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning



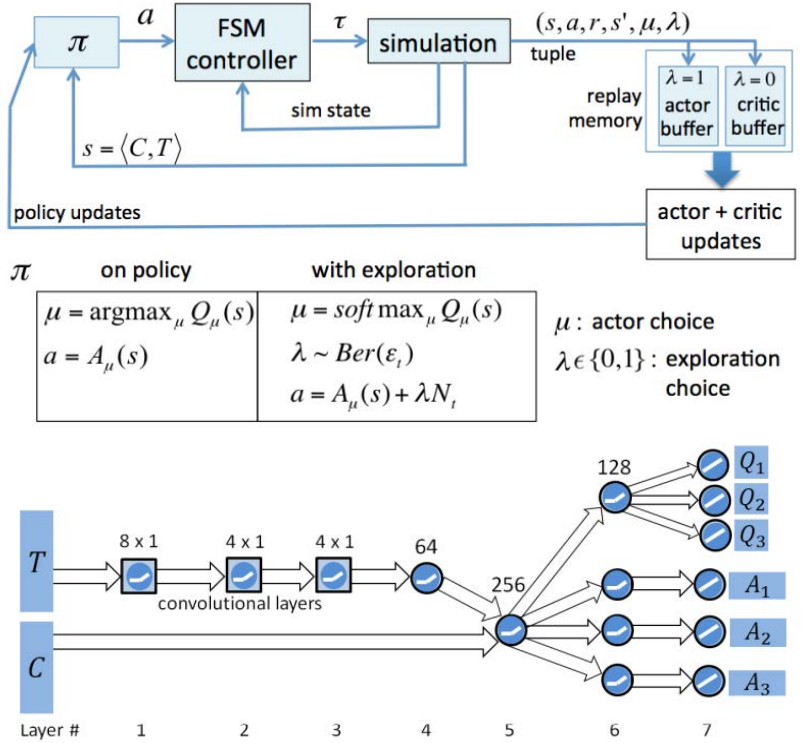
(a) Raptor `mixed` terrain



(b) Raptor `slopes-mixed` terrain



(c) Raptor `narrow-gaps` terrain



Peng X B, Berseth G, van de Panne M. Terrain-adaptive locomotion skills using deep reinforcement learning[J]. ACM Transactions on Graphics (TOG), 2016, 35(4): 81.

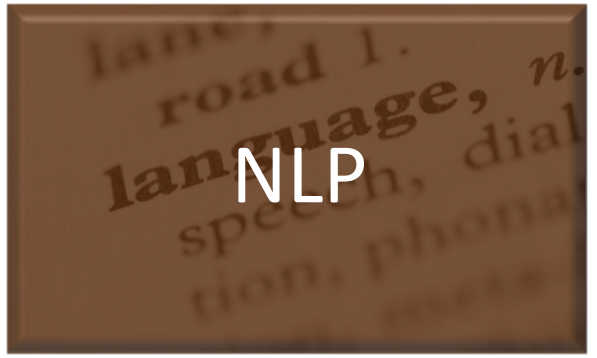

AlphaGo
Game

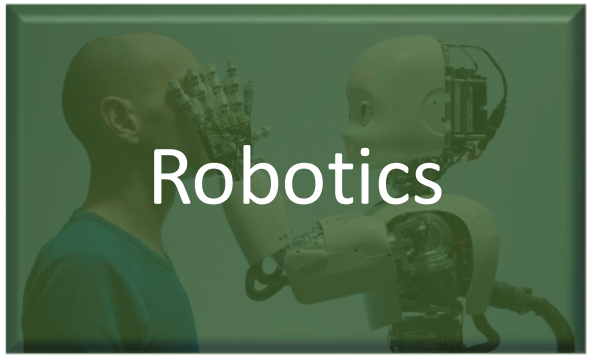

Neuro Science


Music & Movie


Healthcare


Trading


NLP


Robotics


Education

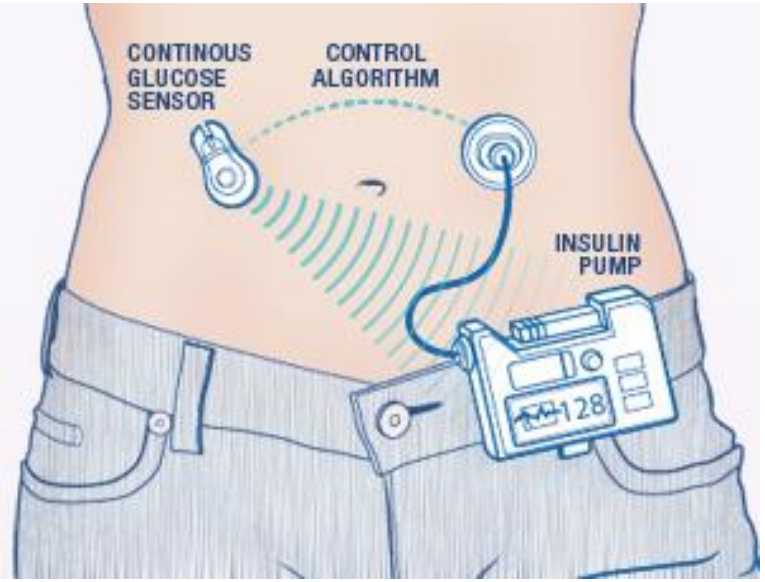

LLM Reasoning

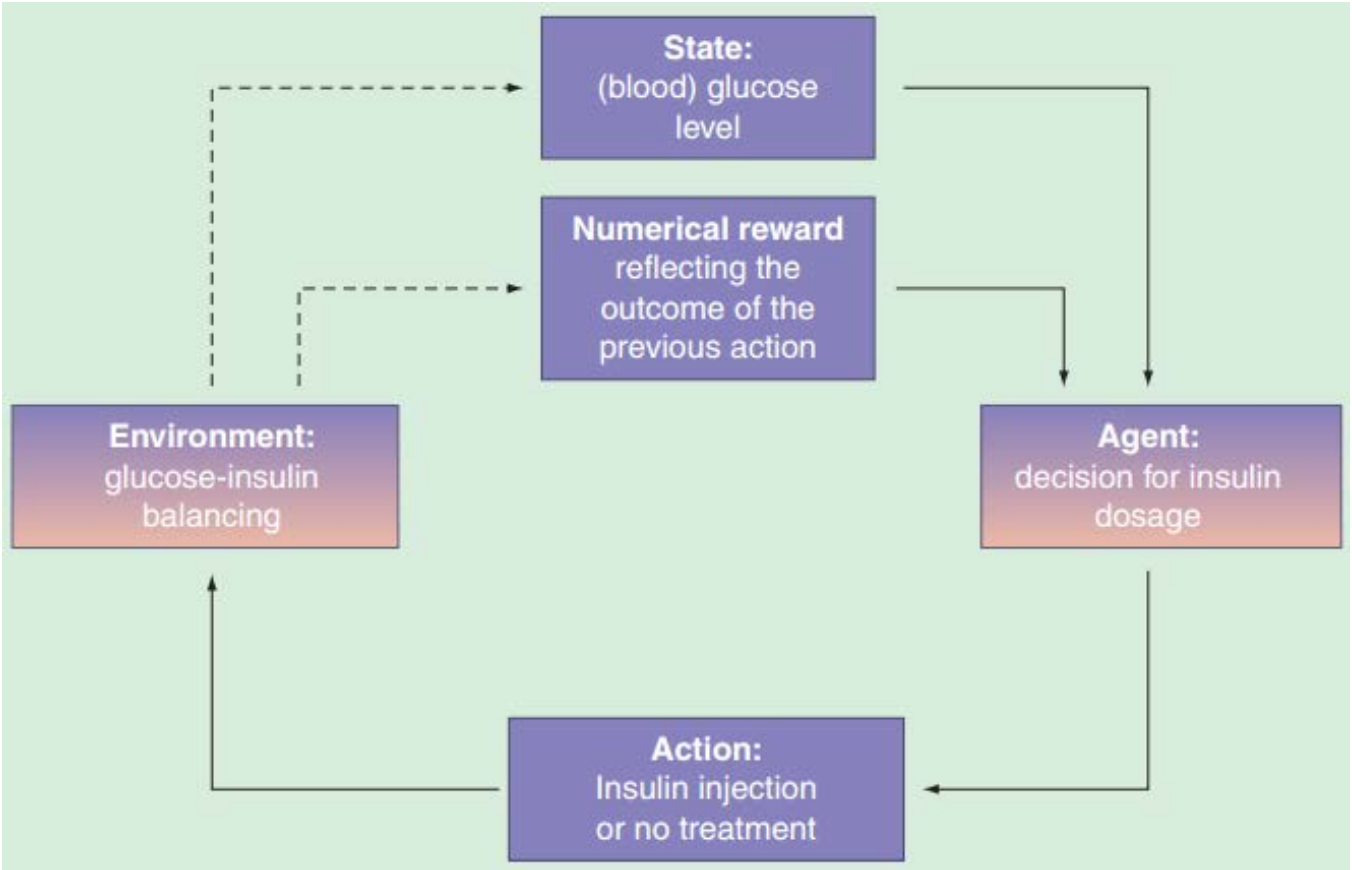
HealthCare

- Sequential Decision Making in HealthCare

HealthCare

- Artificial Pancreas





Bothe M K, Dickens L, Reichel K, et al. The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas[J]. Expert review of medical devices, 2013, 10(5): 661-673.

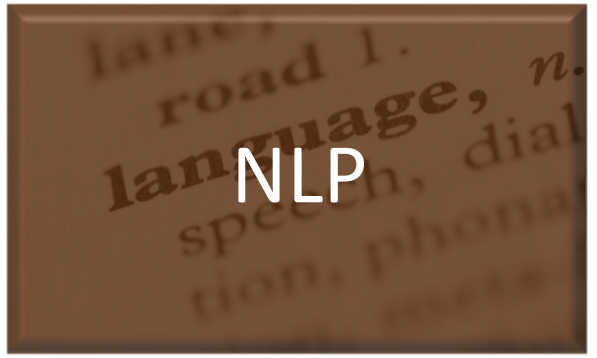

AlphaGo
Game

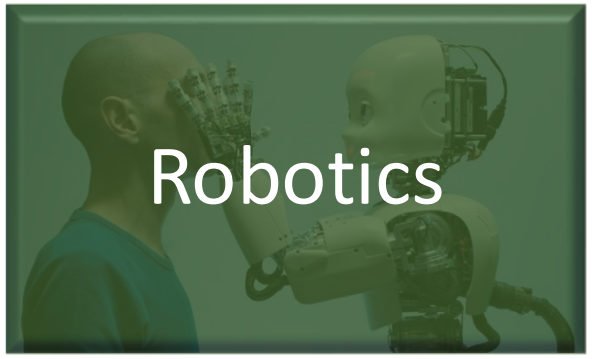

Neuro Science


Music & Movie


Healthcare


Trading


NLP


Robotics


Education


LLM Reasoning

Trading

- Sequential Decision Making in Trading

Trading

- The Success of Recurrent Reinforcement Learning(RRL)
  - Trading systems via RRL significantly outperforms systems trained using supervised methods.
  - RRL-Trader achieves better performance that a Q-Trader for the S&P 500/T-Bill asset allocation problem.
  - Relative to Q-Learning, RRL enables a simple problem representation, avoids Bellman's curse of dimensionality and offers compelling advantages in efficiency.

Learning to Trade via Direct Reinforcement. John Moody and Matthew Saffell, IEEE Transactions on Neural Networks, Vol 12, No 4, July 2001.

Trading

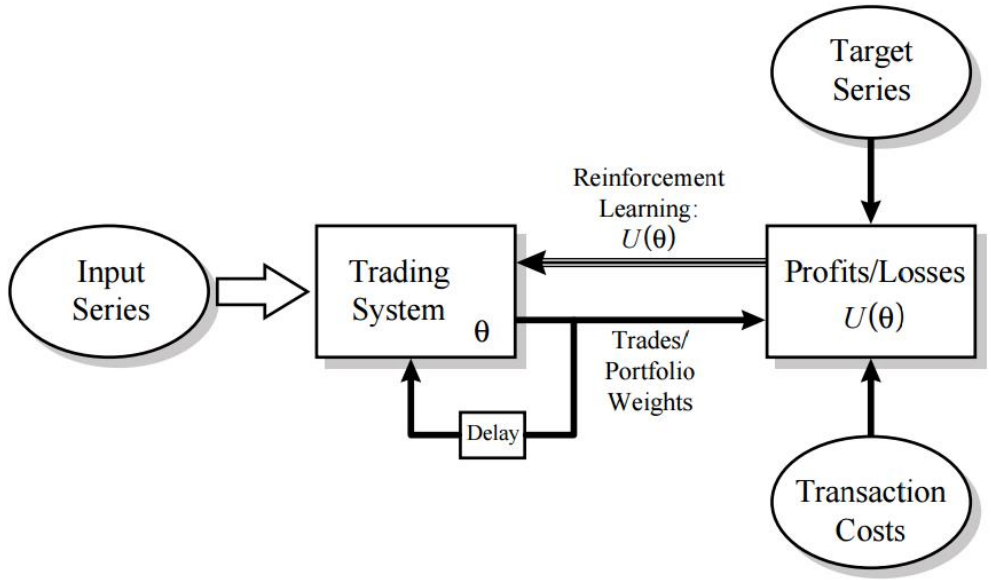
- Special Reward Target for Trading: Sharpe Ratio

$$S_T = \frac{\text{Average}(R_t)}{\text{Standard Deviation}(R_t)}$$

- Recurrent Reinforcement Learning
  - specially tailored policy gradient

$$\frac{dU_t(\theta)}{d\theta_t} \approx \frac{dU_t}{dR_t}\left\{\frac{dR_t}{dF_t}\frac{dF_t}{d\theta_t} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta_{t-1}}\right\}$$

$$\Delta\theta_t = \rho\frac{dU_t(\theta_t)}{d\theta_t}$$



Learning to Trade via Direct Reinforcement. John Moody and Matthew Saffell, IEEE Transactions on Neural Networks, Vol 12, No 4, July 2001.

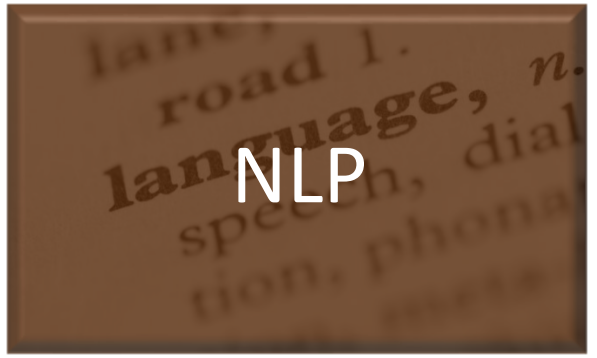

AlphaGo
Game

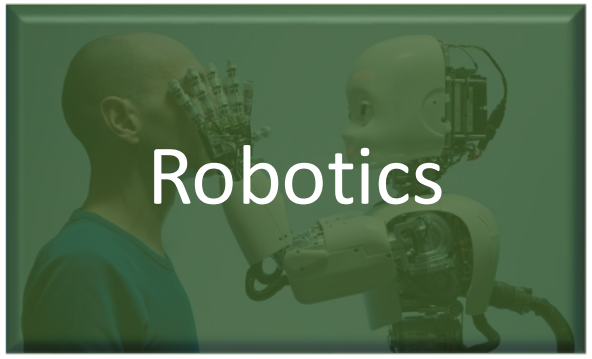

Neuro Science


Music & Movie


Healthcare


Trading


NLP

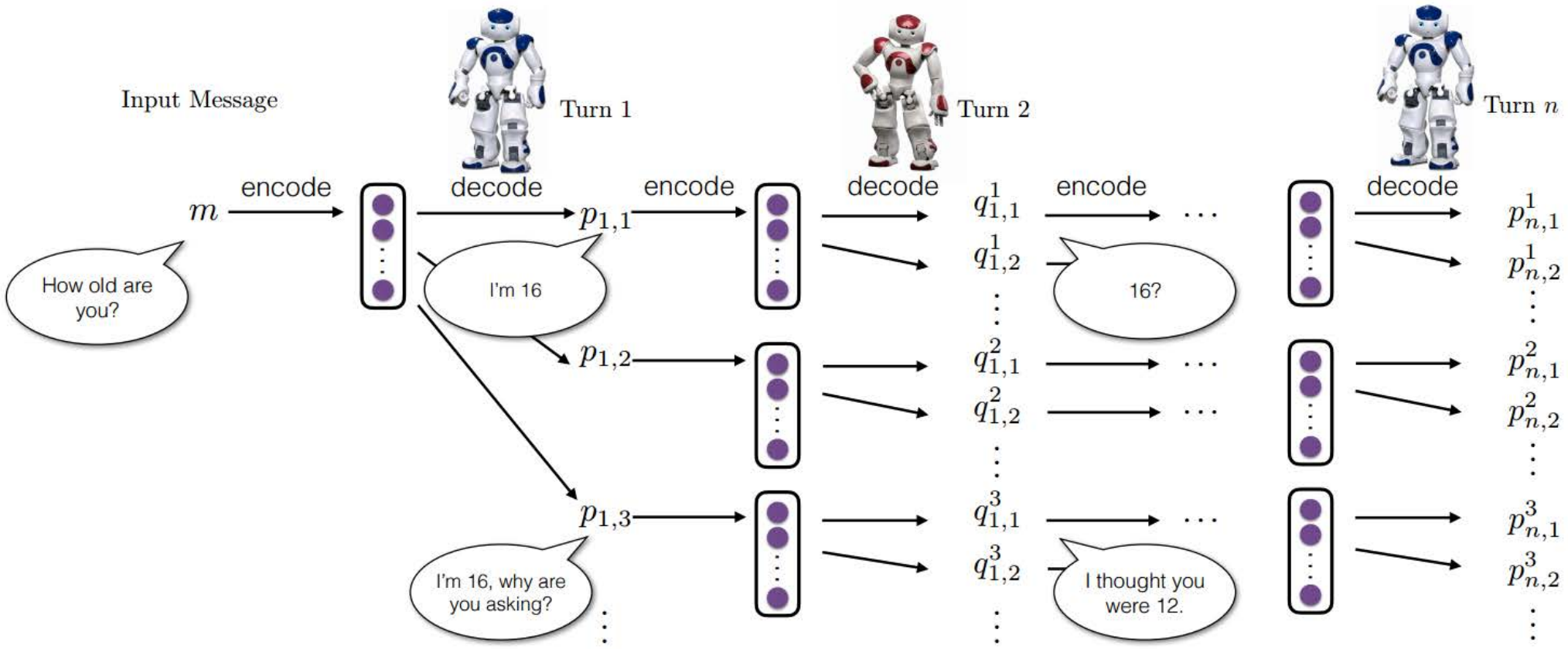

Robotics


Education


LLM Reasoning

Natural Language Processing

- Conversational agents

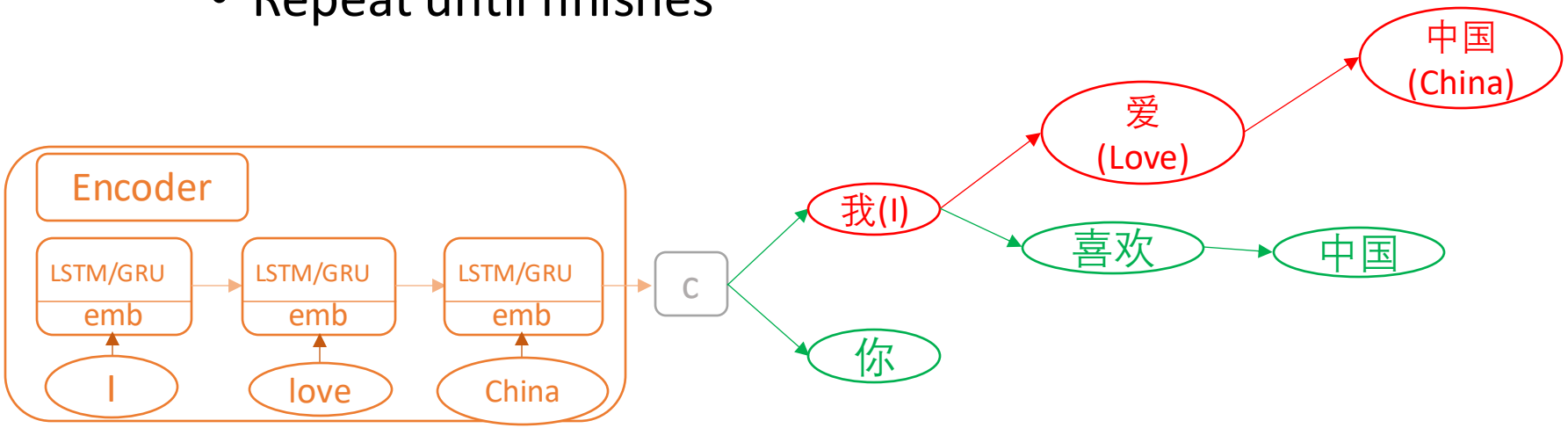


Li J, Monroe W, Ritter A, et al. Deep Reinforcement Learning for Dialogue Generation[J]. arXiv preprint arXiv:1606.01541, 2016.

Baseline mutual information model (Li et al. 2015)	**Proposed reinforcement learning model**
A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6)	A: Where are you going? (1) B: I'm going to the police station. (2) A: I'll come with you. (3) B: No, no, no, no, you're not going anywhere. (4) A: Why? (5) B: I need you to stay here. (6) A: I don't know what you are talking about. (7) ...
A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...	A: How old are you? (1) B: I'm 16. Why are you asking? (2) A I thought you were 12. (3) B: What made you think so? (4) A: I don't know what you are talking about. (5) B: You don't know what you are saying. (6) ... ...

Machine Translation with Value Network

- Decoding with beam search algorithm
  - The algorithm maintain a set of candidates, which are partial sentences
  - Expand each partial sentences by appending a new word
  - Select top-scored new candidates based on the conditional probability P(y|x)
  - Repeat until finishes



Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tie-Yan Liu, Decoding with Value Networks for Neural Machine Translation, NIPS 2017.

Value Network- training and inference

- For each bilingual data pair (x,y), and a translation model from X->Y
  - Use the model to sample a partial sentence y_p with random early stop
  - Estimate the expected BLEU score on (x, y_p)
  - Learn the value function based on the generated data
- Inference : similar to AlphaGo

$$\frac{1}{|y|}\log P(y|x) \quad \Longrightarrow \quad \alpha \times \frac{1}{|y|}\log P(y|x) + (1-\alpha) \times \log v(x,y),$$

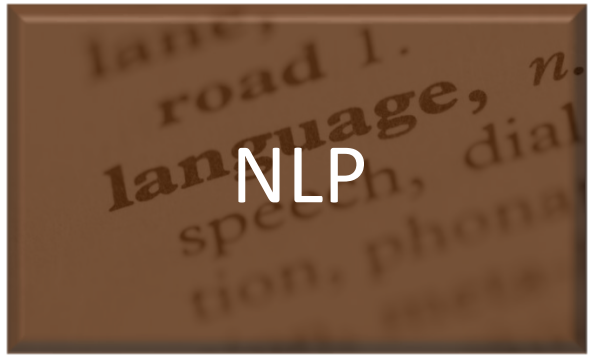

AlphaGo
Game

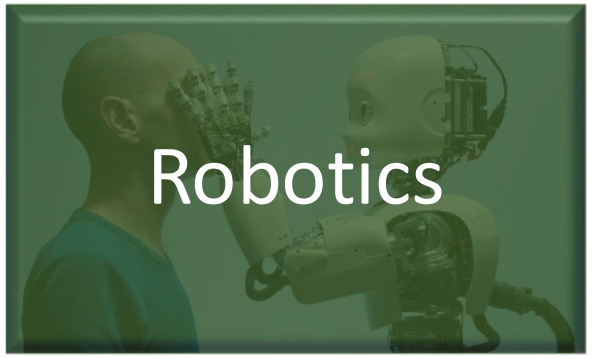

Neuro Science


Music & Movie


Healthcare


Trading


NLP


Robotics


Education

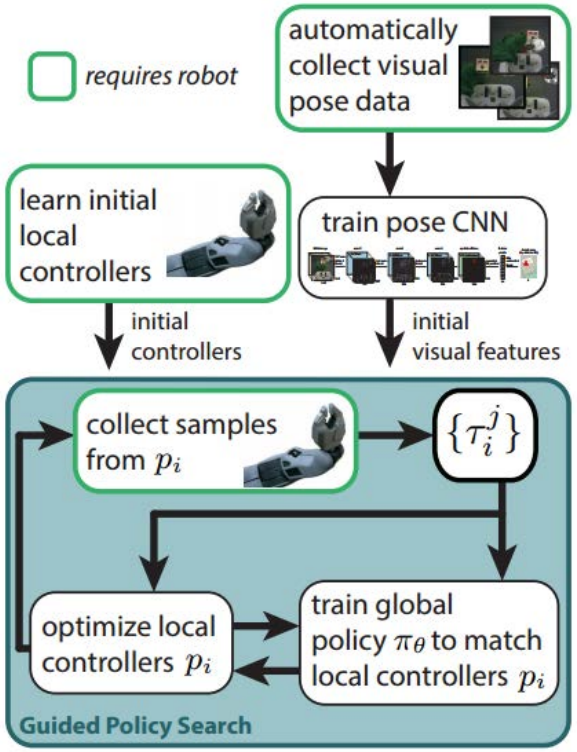

LLM Reasoning

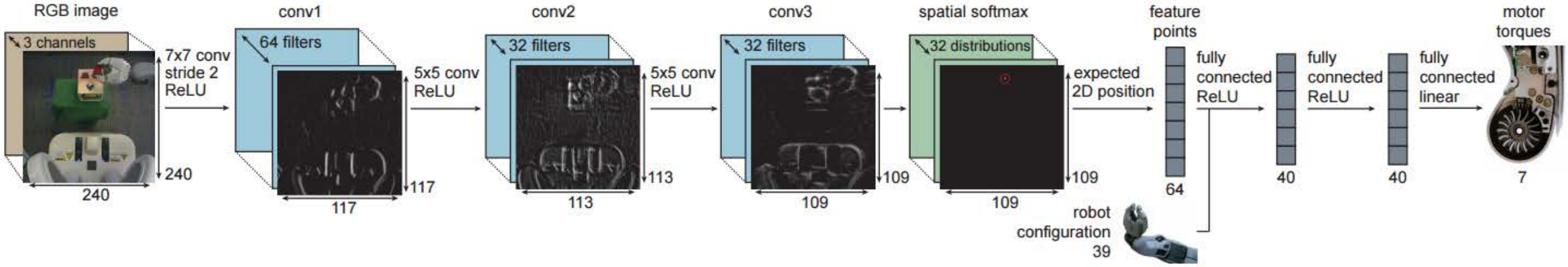
Robotics/Embodied AI

- Sequential Decision Making in Robotics

Robotics /Embodied AI

- End-to-End Training of Deep Visuomotor Policies





Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies[J]. Journal of Machine Learning Research, 2016, 17(39): 1-40.

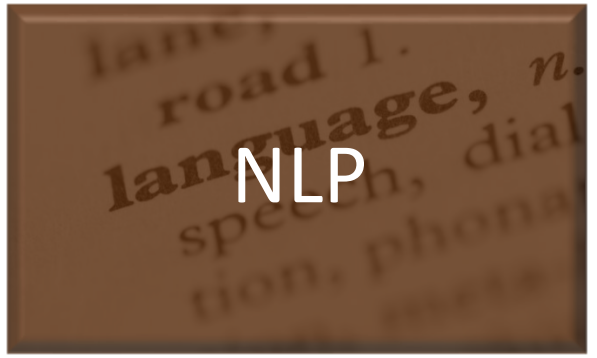

AlphaGo
Game

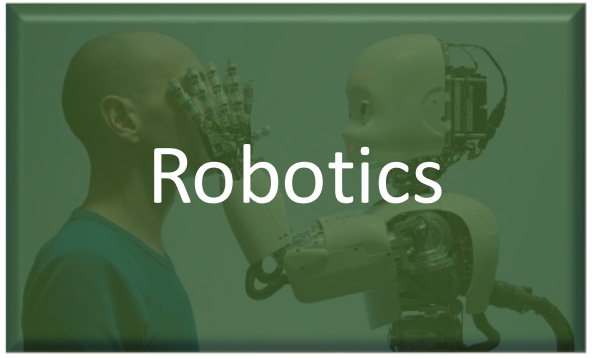

Neuro Science


Music & Movie


Healthcare


Trading


NLP


Robotics


Education

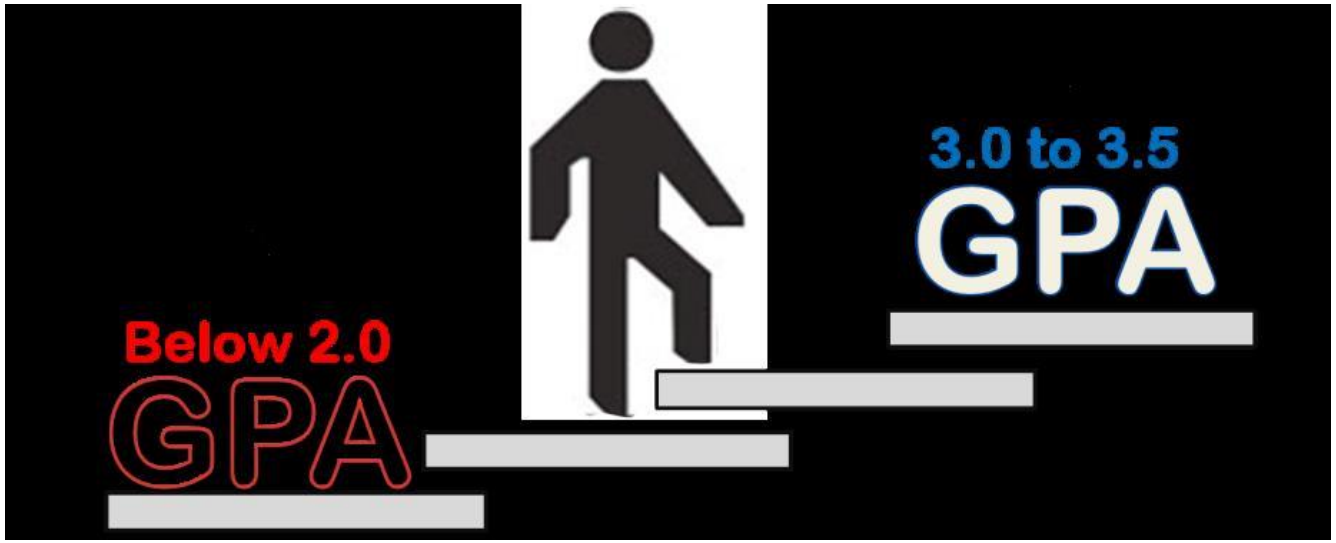

LLM Reasoning

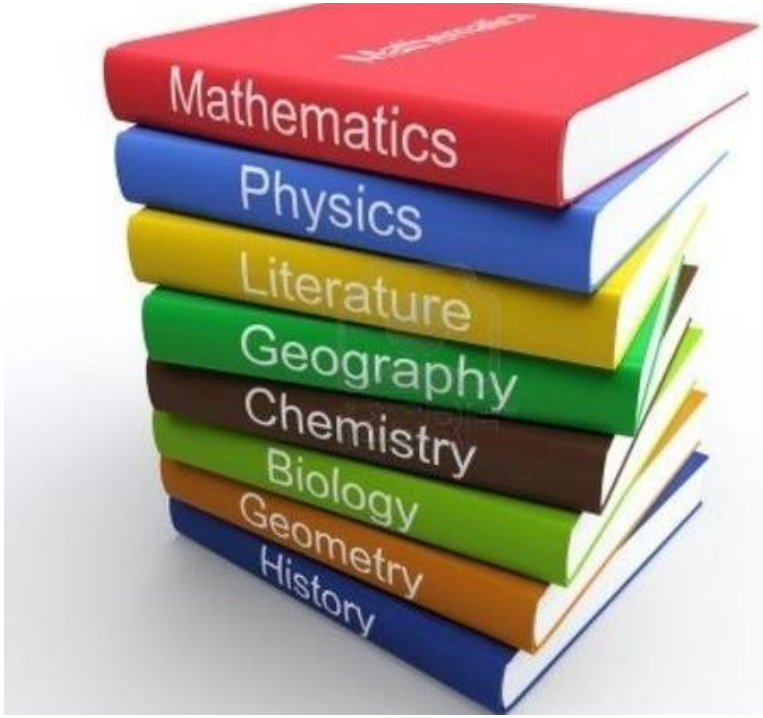
Education

- Agents making decisions as interact with students
- Towards efficient learning

Education

- Personalized curriculum design
  - Given the diversity of students knowledge, learning behavior, and goals.
  - Reward: get the highest cumulative grade





Hoiles W, Schaar M. Bounded Off-Policy Evaluation with Missing Data for Course Recommendation and Curriculum Design[C]//Proceedings of The 33rd International Conference on Machine Learning. 2016: 1596-1604.

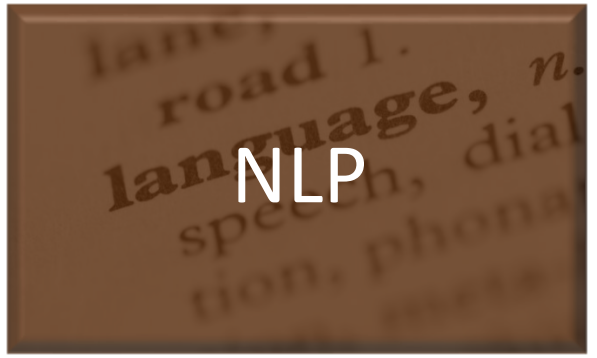

AlphaGo
Game

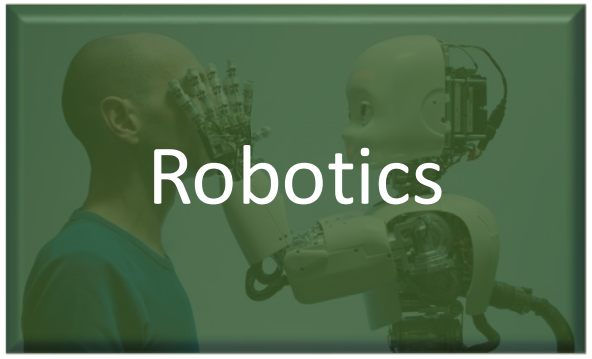

Neuro Science


Music & Movie


Healthcare


Trading

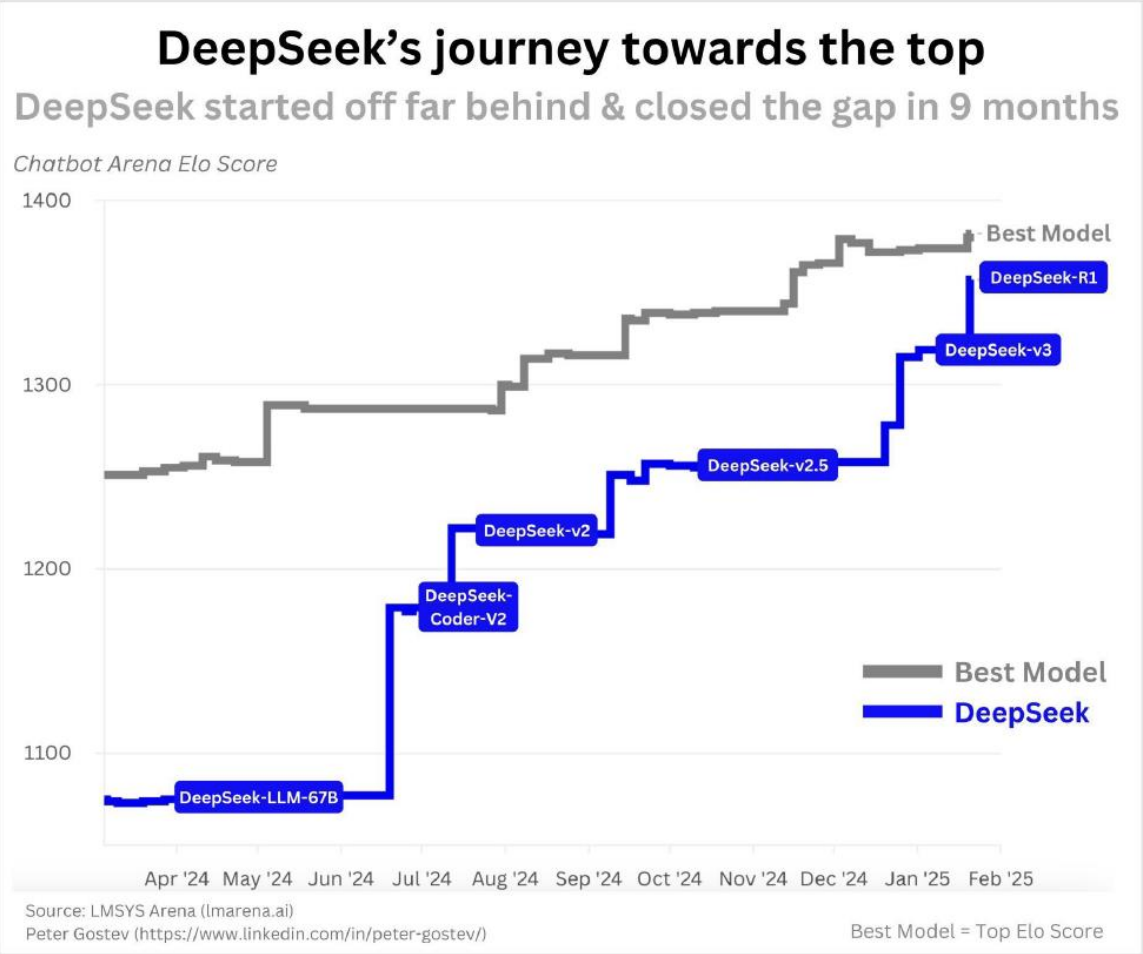

NLP


Robotics

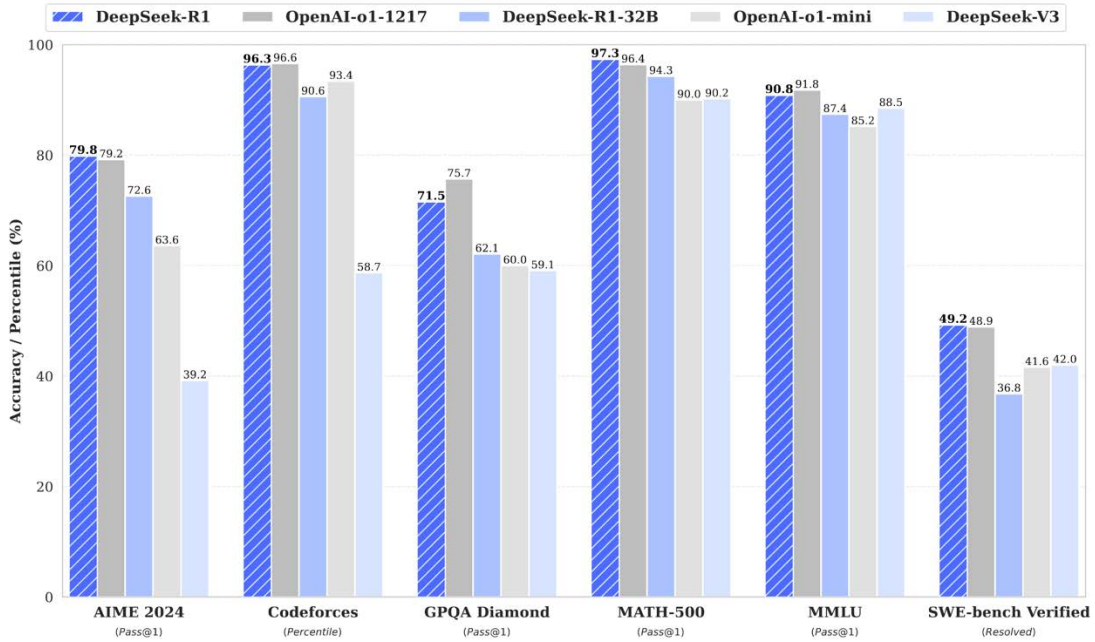

Education


LLM Reasoning

LLM reasoning

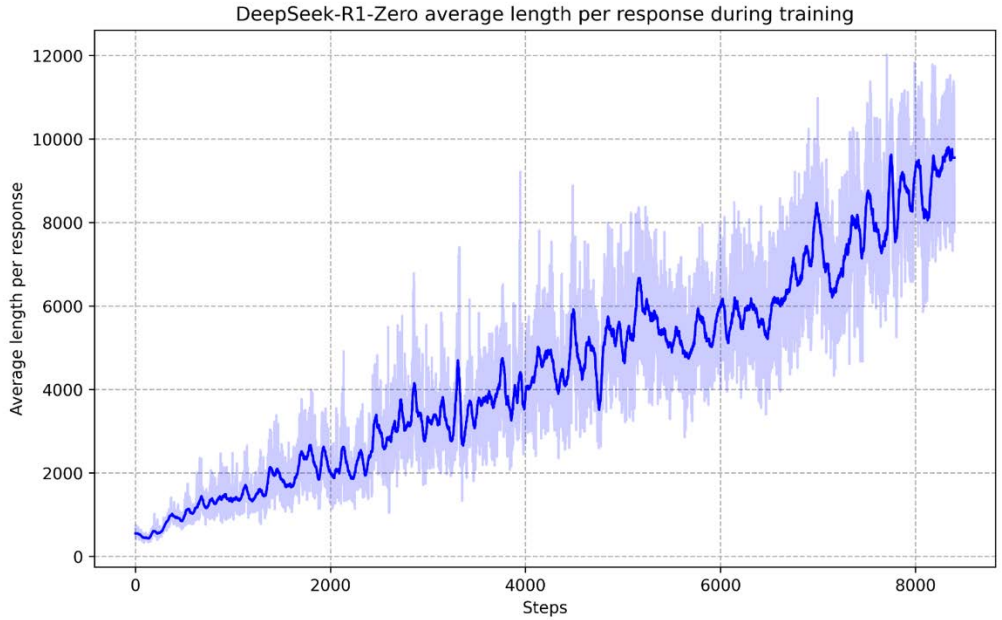


RL finetuning plays a key role

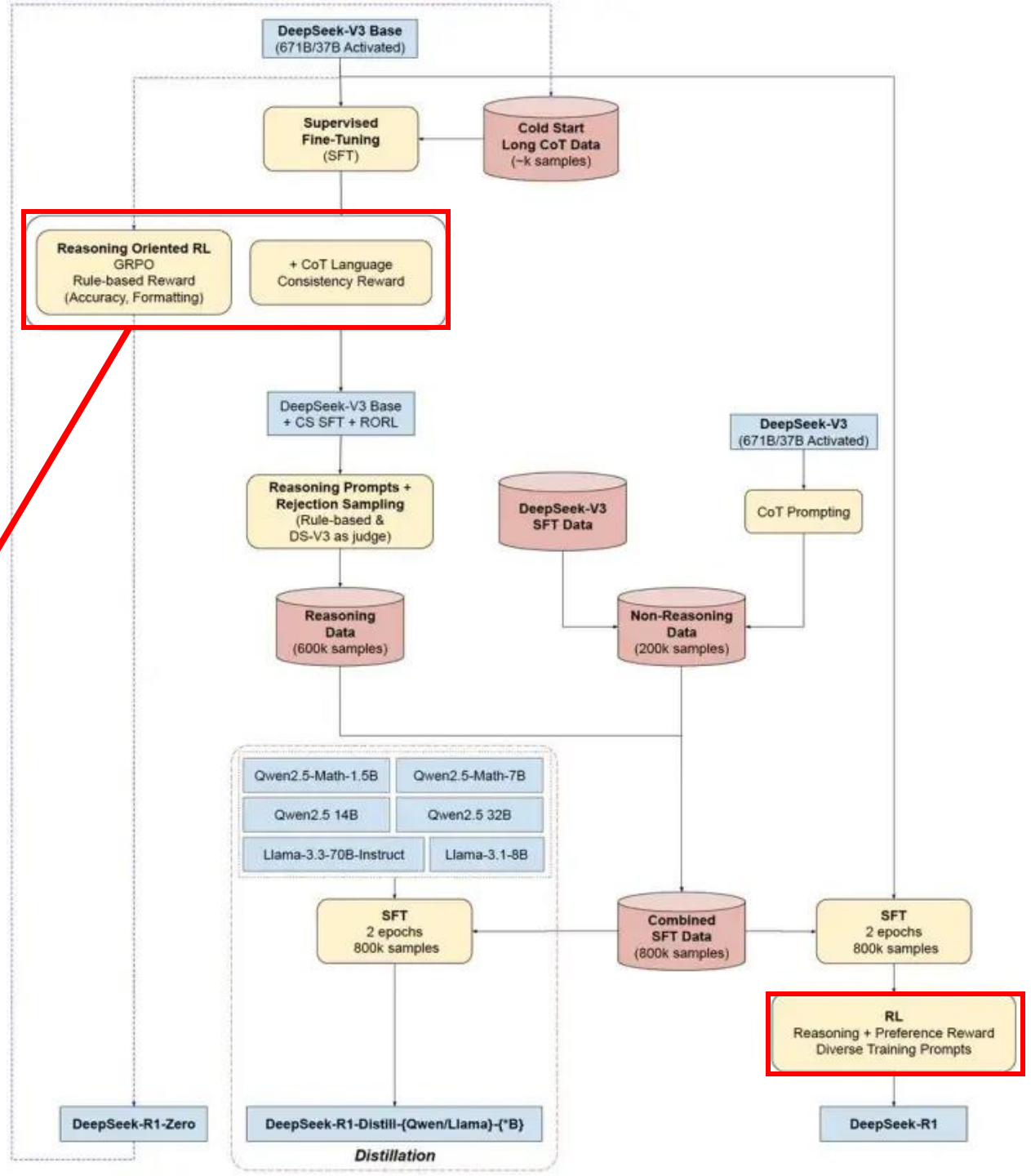


[1] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025.

LLM reasoning

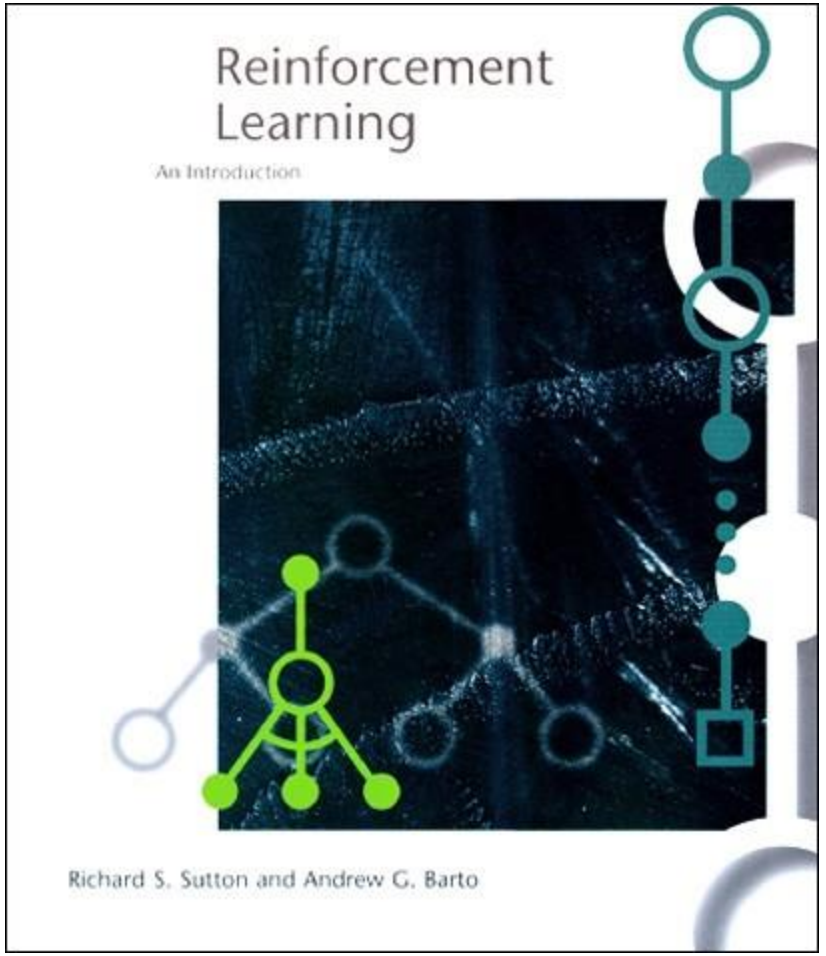


Self-emergence of long-chain reasoning behavior during RL finetuning



References

- Recent progress
  - NIPS, ICML, ICLR
  - AAAI, IJCAI
- Courses
  - Reinforcement Learning, David Silver, with videos http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html
  - Deep Reinforcement Learning, Sergey Levine, with videos http://rll.berkeley.edu/deeprlcourse/
- Textbook
  - Reinforcement Learning: An Introduction, Second edition,Richard S. Sutton and Andrew G. Barto http://www.incompleteideas.net/book/the-book-2nd.html



Acknowledgements

- Some content borrowed from David Silver’s lecture
- My colleagues Li Zhao, Di He
- My student Yu Zheng, Zefang Zong, Qianyue Hao.

Thanks!

lizo@microsoft.com

yuwang5@microsoft.com

liyong07@tsinghua.edu.cn