

强化学习简介

秦涛

微软亚洲研究院

Outline

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

General Introduction

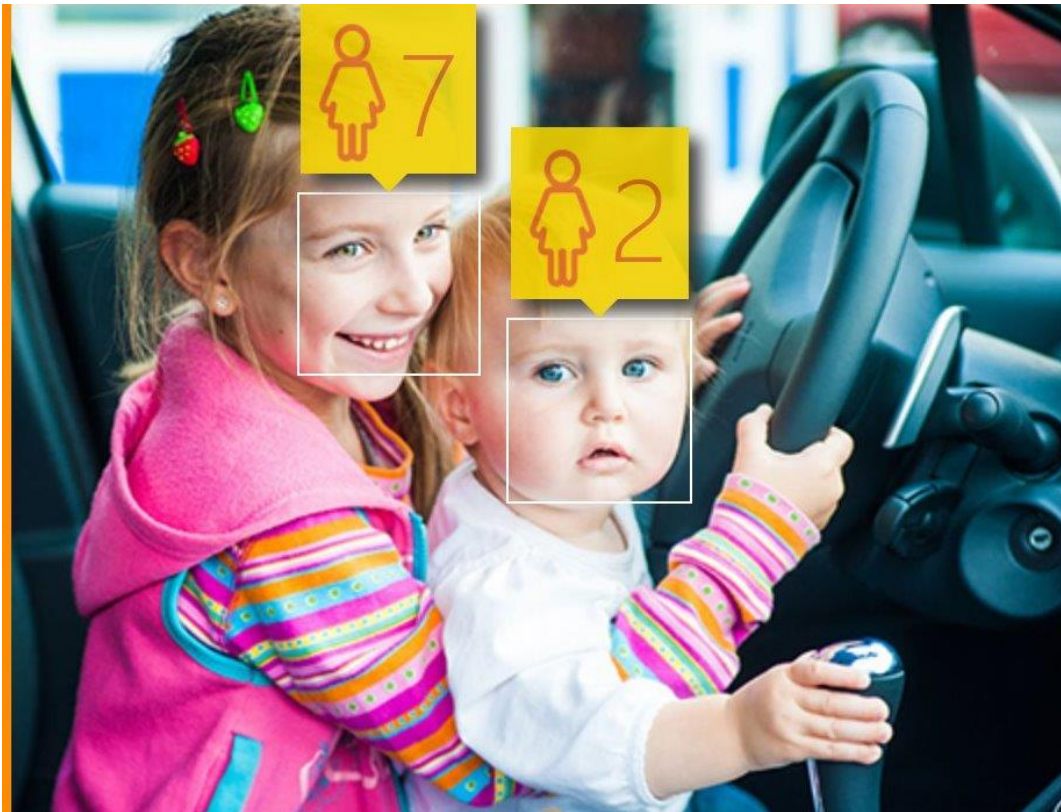
Machine Learning

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

ccl 2018

网页 图片 视频 学术 词典 地图

检测到您输入了英文，试试切换到国际版？搜英文结果更丰富更准确 >

5,990,000 条结果 时间不限 ▾

第十七届中国计算语言学大会 (CCL 2018) 及第六届基于 ...
2018-9-18 · “第十七届中国计算语言学大会” (The Seventeenth China National Conference on Computational Linguistics, CCL 2018) 将于2018年10月19日—21日在长沙理工 ...
www.cips-cl.org/static/CCL2018/index.html ▾

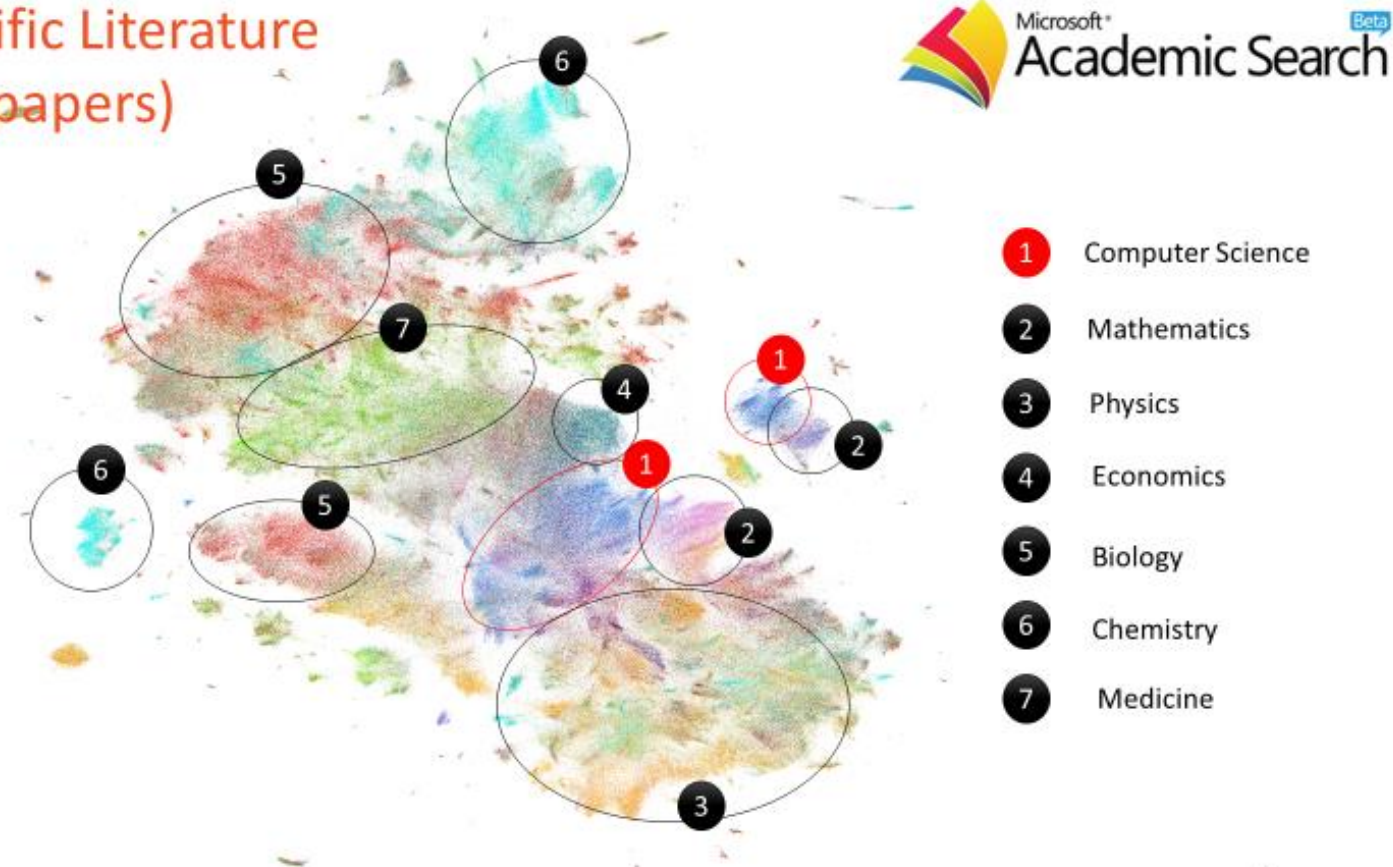
Leadership Development Results That Matter | CCL 翻译此页
2018-10-11 · Discover CCL's global leadership development research and solutions for you, your team, your business and the world.
<https://www.ccl.org> ▾

2018 CONCACAF Champions League - Wikipedia 翻译此页 | 中文网页
Runners-up: [Toronto FC](#) Champions: [Guadalajara](#) (2nd title)
Dates: February 20 – April 25, 2018 Teams: 16 (from 8 associations)
2018-10-6 · The **2018** CONCACAF Champions League (officially the **2018** Scotiabank CONCACAF Champions League for sponsorship reasons) was the 10th edition of the CONCACAF Champions League under its current name, and overall the 53rd edition of the premier football club competition organized by CONCACAF, the regional governing body of North America, Central ...
https://en.wikipedia.org/wiki/2018_CONCACAF_Champions_League ▾

Unsupervised Learning

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

Scientific Literature
(10M papers)



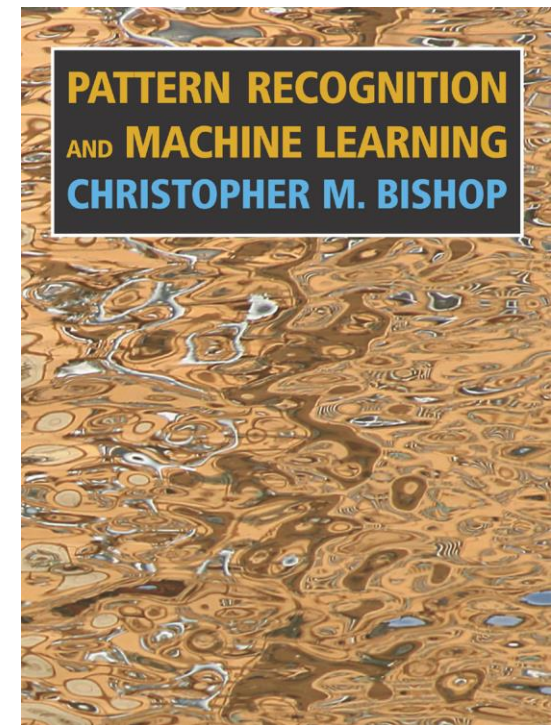
Reinforcement Learning



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....

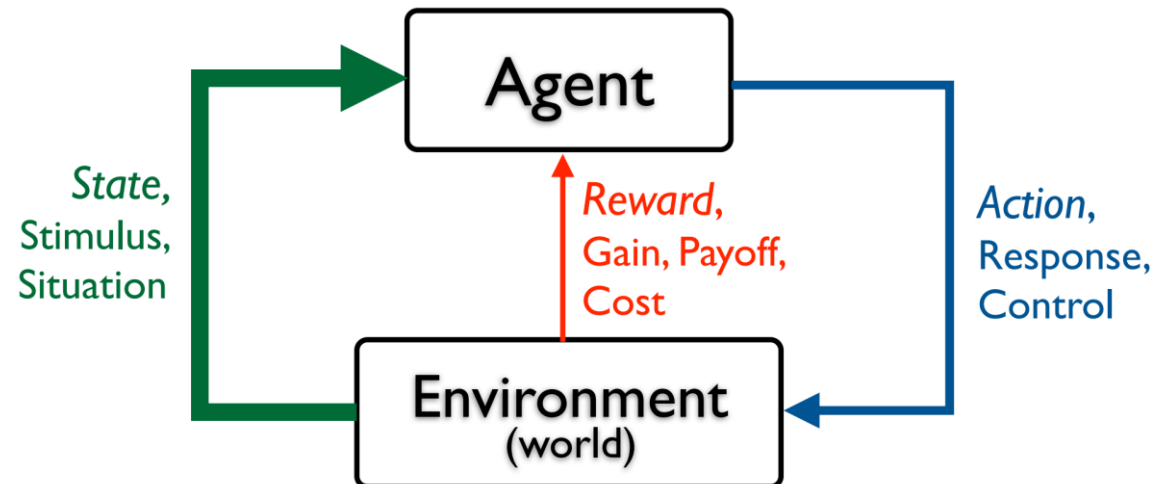


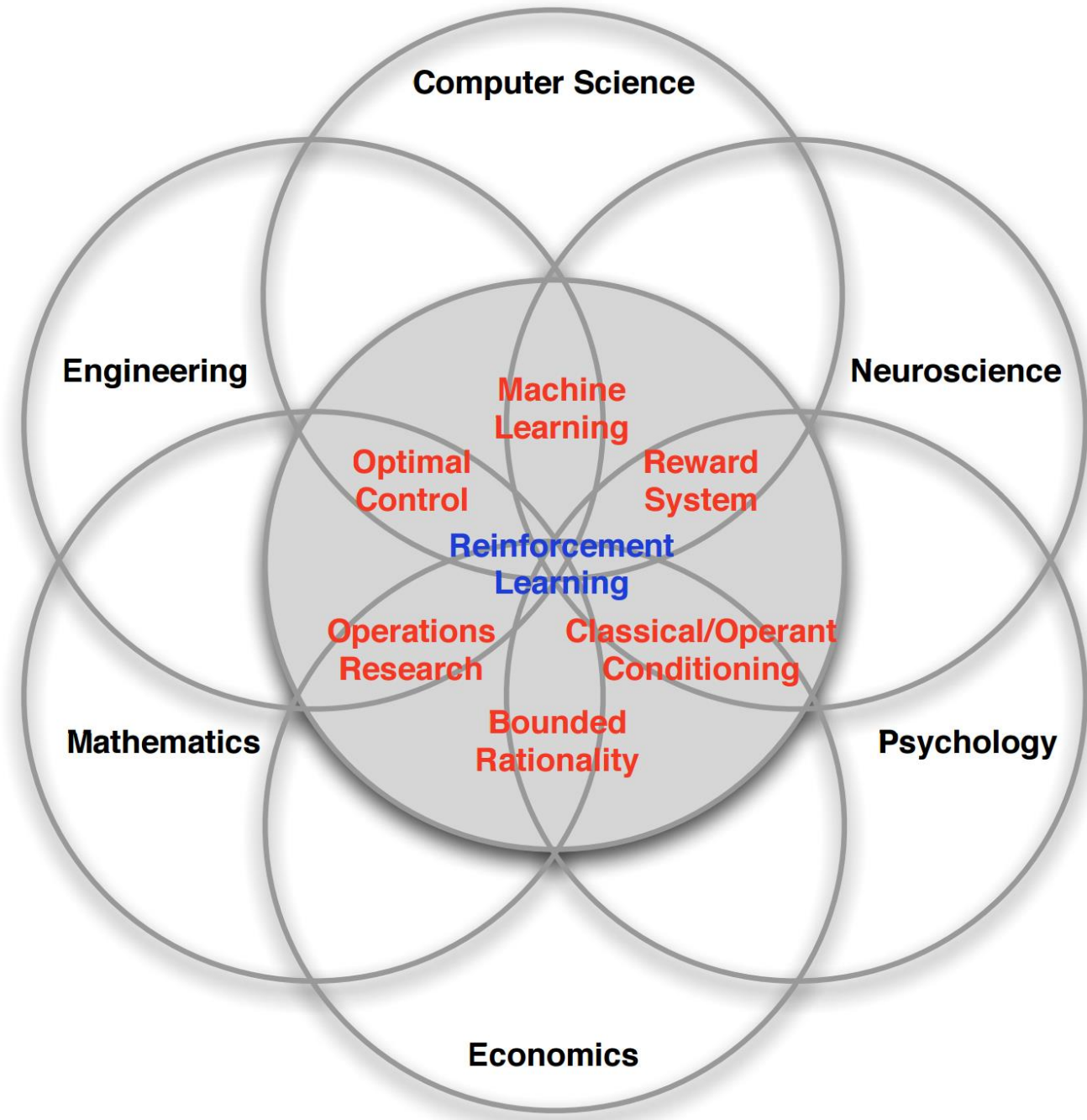
Reinforcement learning problems involve learning **what to do - how to map situations to actions** - so as to maximize a numerical reward signal.



Reinforcement Learning

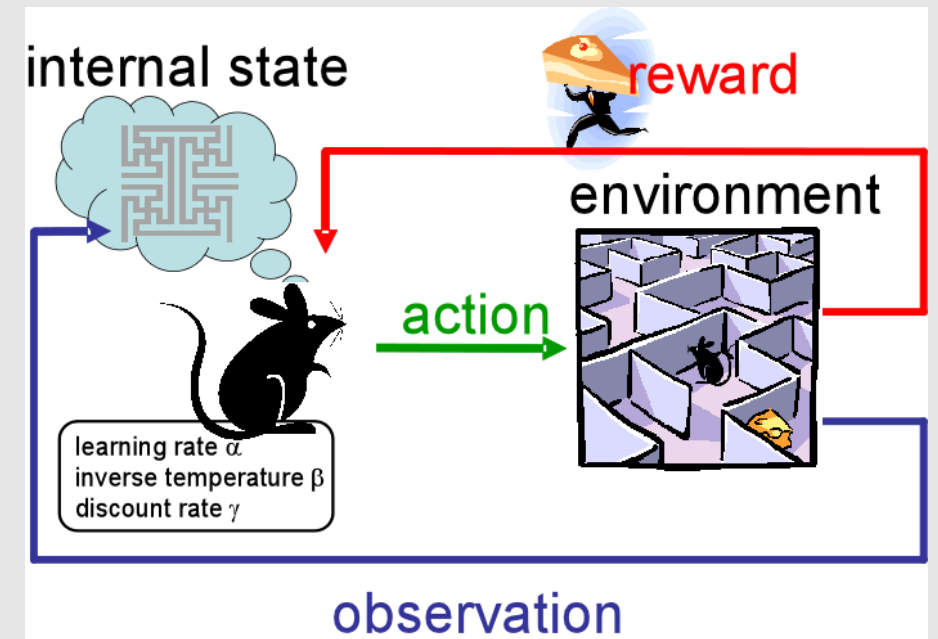
- Agent-oriented learning-learning by interacting with an environment to achieve a goal
 - Learning by trial and error, with only delayed evaluative feedback(reward)
 - Agent learns a policy mapping states to actions
 - Seeking to maximize its cumulative reward in the long run





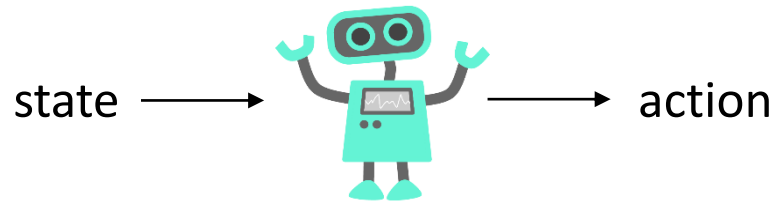
RL vs Other Machine Learning

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



One-shot Decision v.s Sequential Decisions

- Agent Learns a Policy

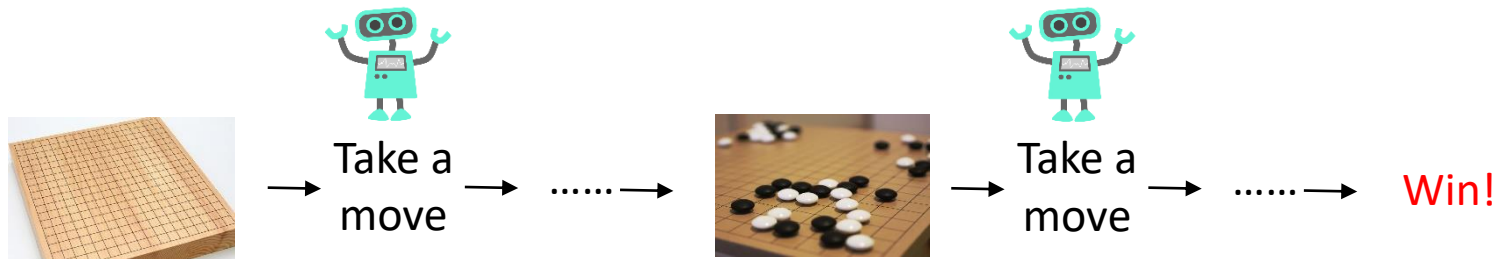


- Supervised Learning

$$f(\text{audio waveform}) = \text{"How are you"}$$

$$f(\text{cat image}) = \text{"Cat"}$$

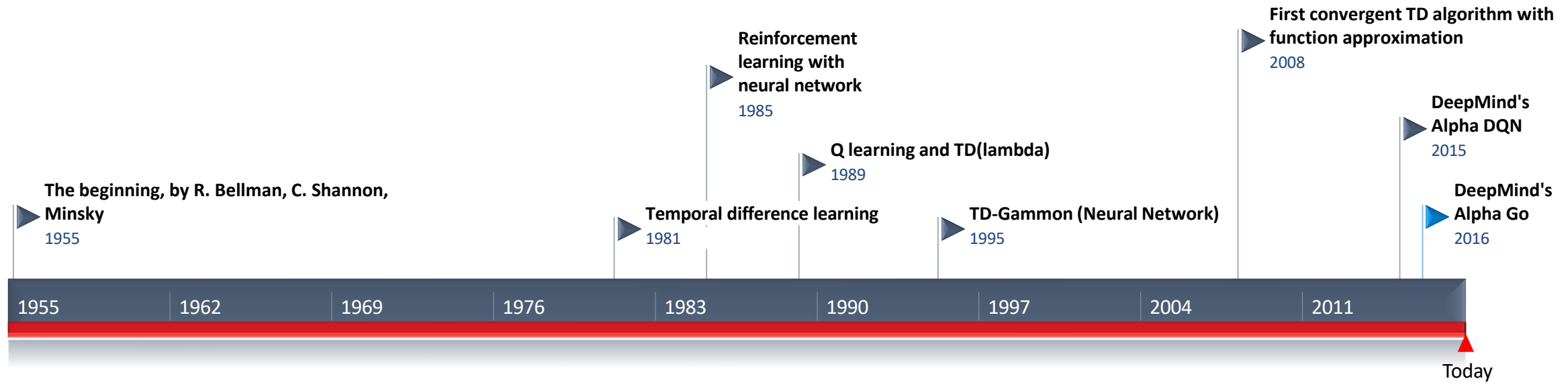
- Reinforcement Learning



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

1955

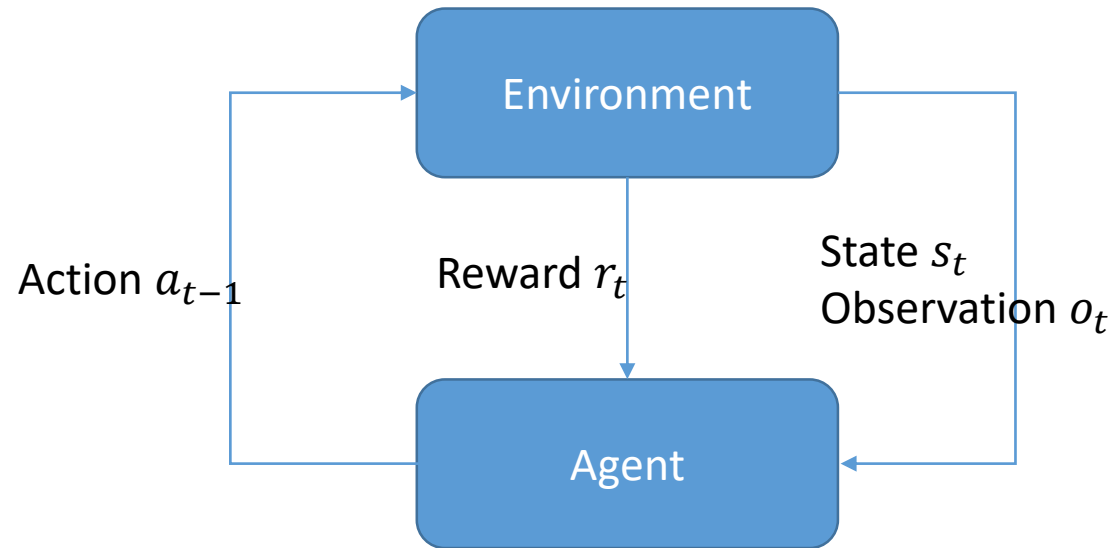




RL has achieved a wide of success across different applications.

Basic Settings

Reinforcement Learning

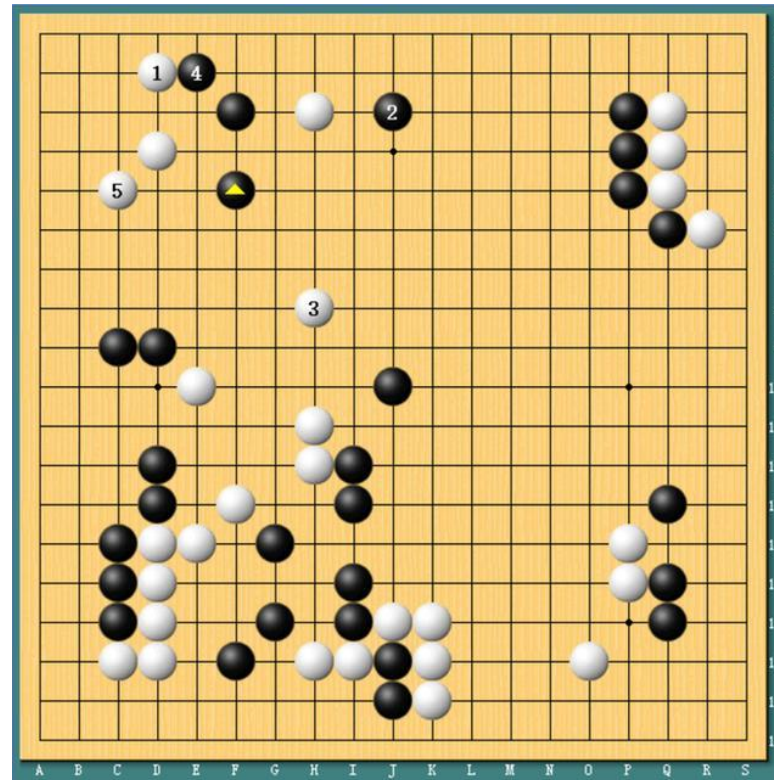


- ❑ a set of environment states S ;
- ❑ a set of actions A ;
- ❑ rules of transitioning between states;
- ❑ rules that determine the scalar immediate reward of a transition; and
- ❑ rules that describe what the agent observes.

Goal: Maximize expected long-term payoff

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 |



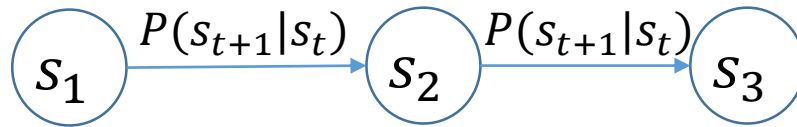
Example Applications

| 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 Chain

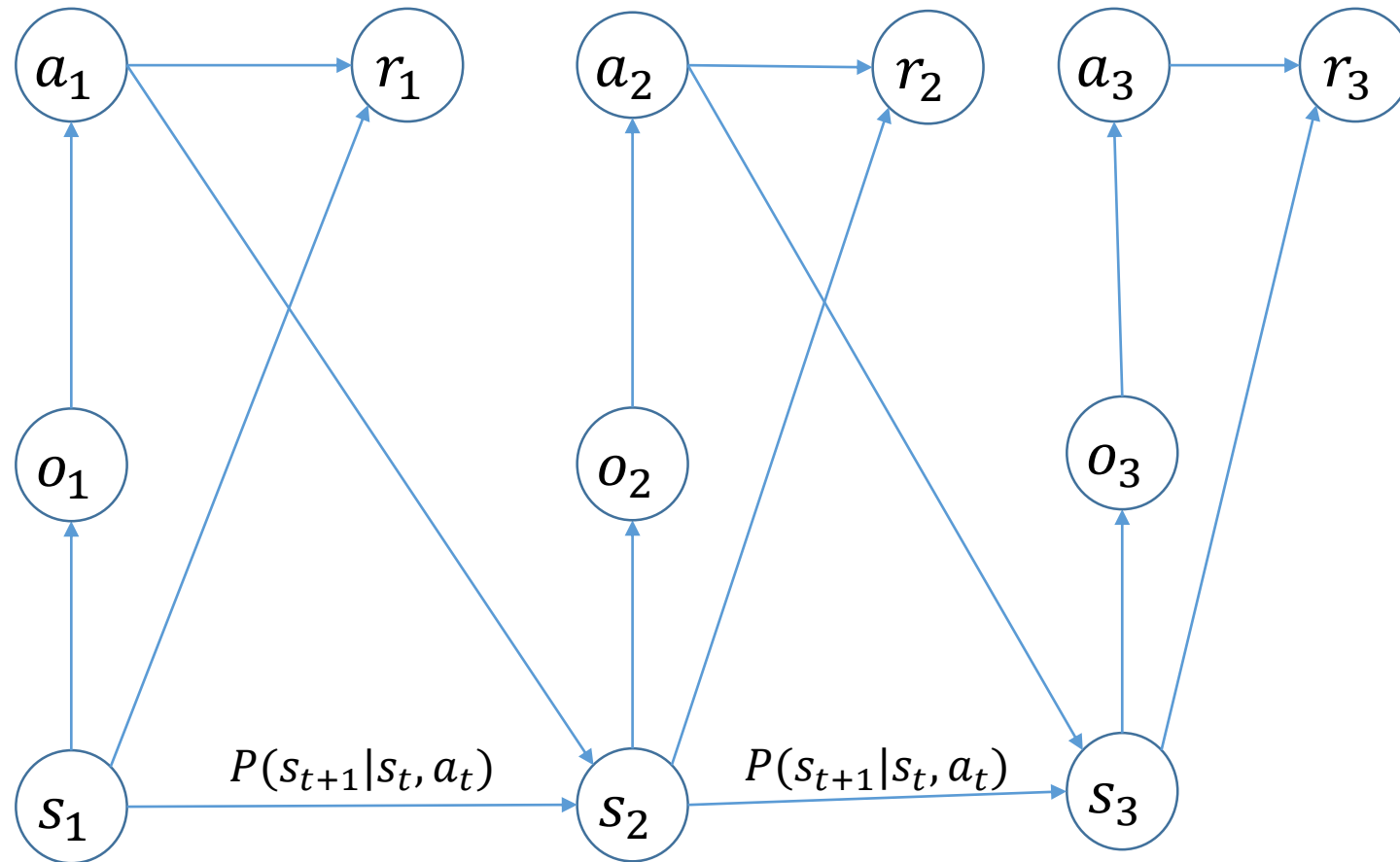
- Markov state

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



Andrey Markov

Markov Decision Process



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

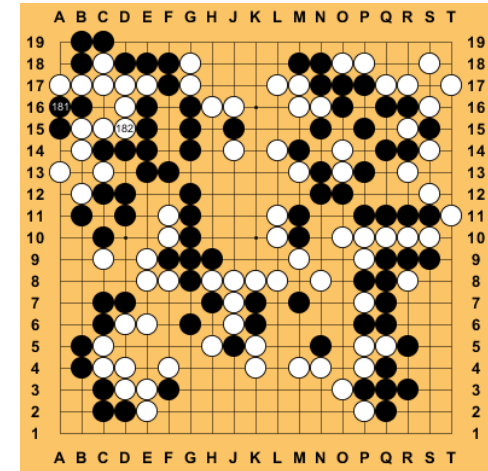
Markov Decision Process

- 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: $(S, A, \mathcal{P}, \mathcal{R}, \gamma)$

- S is a finite set of states
- 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.

- $\dots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \dots$

Policy

- A mapping from state to action

- Deterministic $a = \pi(s)$
- Stochastic $p = \pi(s, a)$

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

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

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:

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



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:

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



Delayed reward is taken into consideration.

- Action-value functions decompose into Bellman expectation equation.

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

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.

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

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
 - $v_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s]$
 - $q_\pi(s, a) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s, a_t = a]$

Tabular Approaches

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

$$\begin{aligned}v_{\pi}(s) &= E_{\pi}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s\} \\ &= E_{\pi}[r_{t+1} + \gamma v_{\pi}(s') | s] \\ &= r(s, \pi(s)) + \gamma \sum_{s'} P(s' | s, \pi(s)) v_{\pi}(s')\end{aligned}$$

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

- Action-value function

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



Richard Bellman

Planning (Policy Evaluation)

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

Algorithm:

Arbitrary initialization: v_0

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

Stopping criterion:
$$v_{\pi}^{k+1} = r_{\pi} + \gamma P_{\pi} v_{\pi}^k$$
$$|v_{\pi}^{k+1} - v_{\pi}^k| \leq \epsilon$$

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 (Optimal Control)

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

Value iteration with Bellman optimality equation :

Arbitrary initialization: q_0

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

$$\forall s \in S, a \in A \quad q_{k+1}(s, a) = r(s, a) + \gamma \sum_{s' \in S} P(s' | s, a) \max_{a'} q_k(s', a')$$

Stopping criterion: $\max_{s \in S, a \in A} |q_{k+1}(s, a) - q_k(s, a)| \leq \epsilon$

Learning in MDPs

- Have access to the real system but no model
- Generate experience $o_1, a_1, r_1, o_2, a_2, r_2, \dots, o_{t-1}, a_{t-1}, r_{t-1}, o_t$
- Two kinds of approaches
 - Model-free learning
 - Model-based learning

Monte-Carlo Policy Evaluation

- To evaluate state s
- The **first** time-step t that state s is visited in an episode,
- Increment counter $N(s) = N(s) + 1$
- Increment total return $S(s) = S(s) + G_t$
- Value is estimated by mean return $V(s) = \frac{S(s)}{N(s)}$
- By law of large numbers, $V(s) \rightarrow v_{\pi}(s)$ as $N(s) \rightarrow \infty$

Incremental Monte-Carlo Update

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

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 Reinforcement Learning

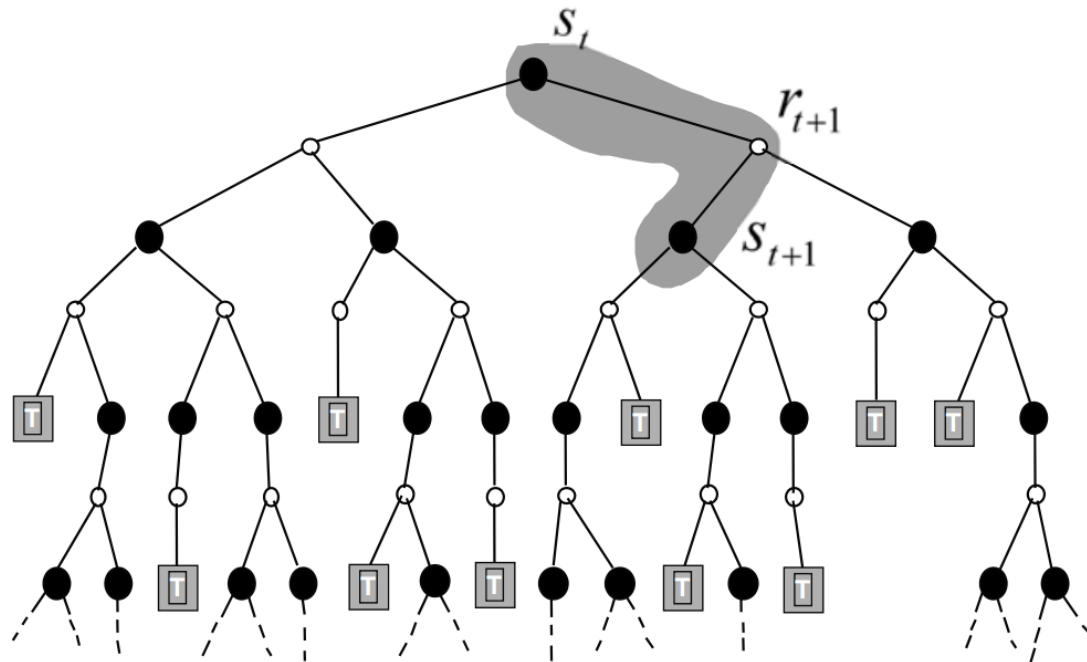
- MC methods learn directly from episodes of experience
- MC is model-free: no knowledge of MDP transitions / rewards
- MC learns from complete episodes
 - Values for each state or pair state-action are updated only based on final reward, not on estimations of neighbor states
- MC uses the simplest possible 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



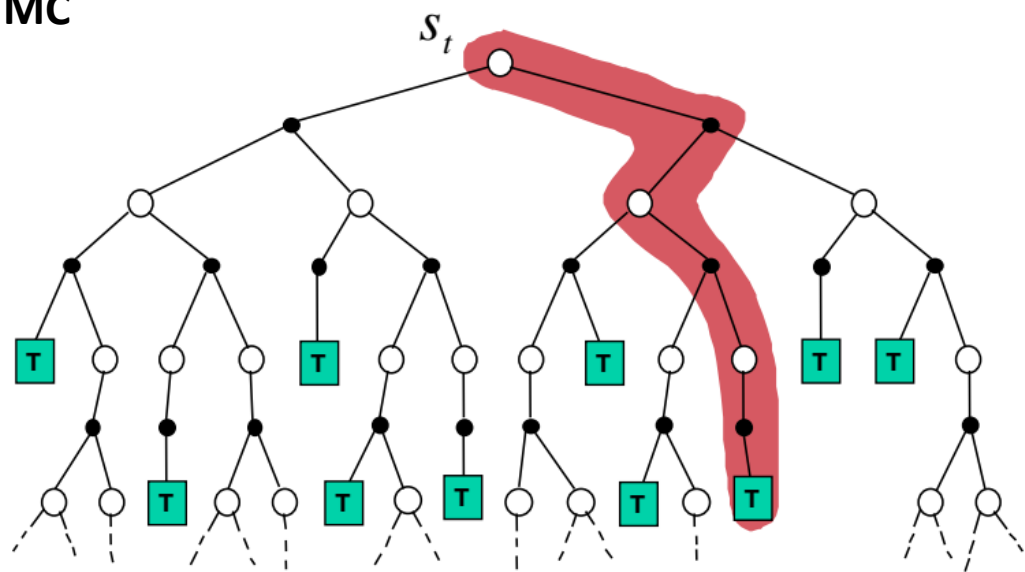
Temporal-Difference Policy Evaluation

- 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

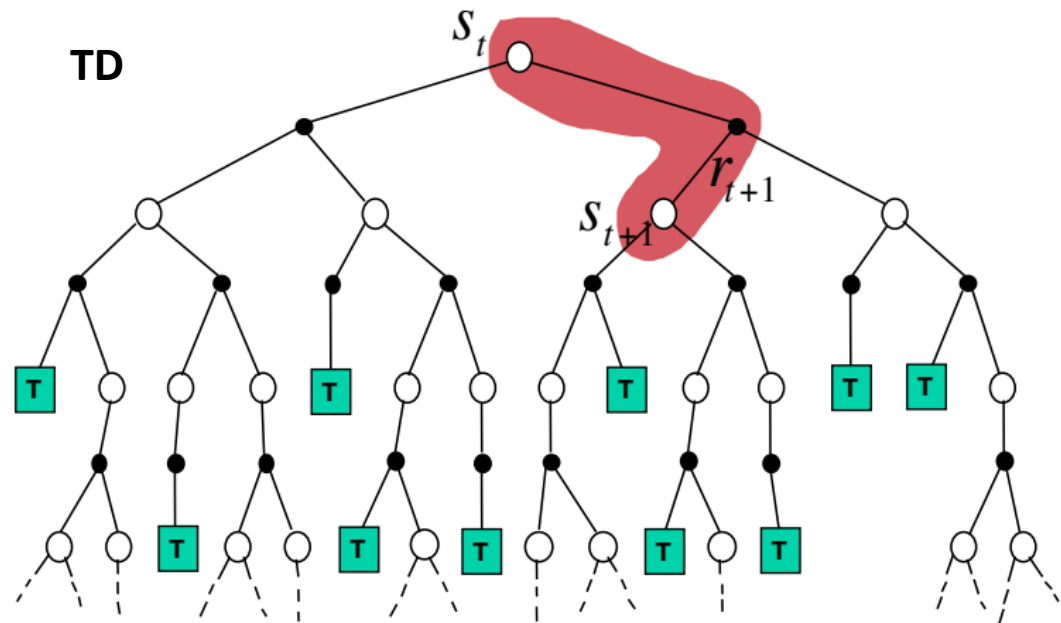
$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

MC



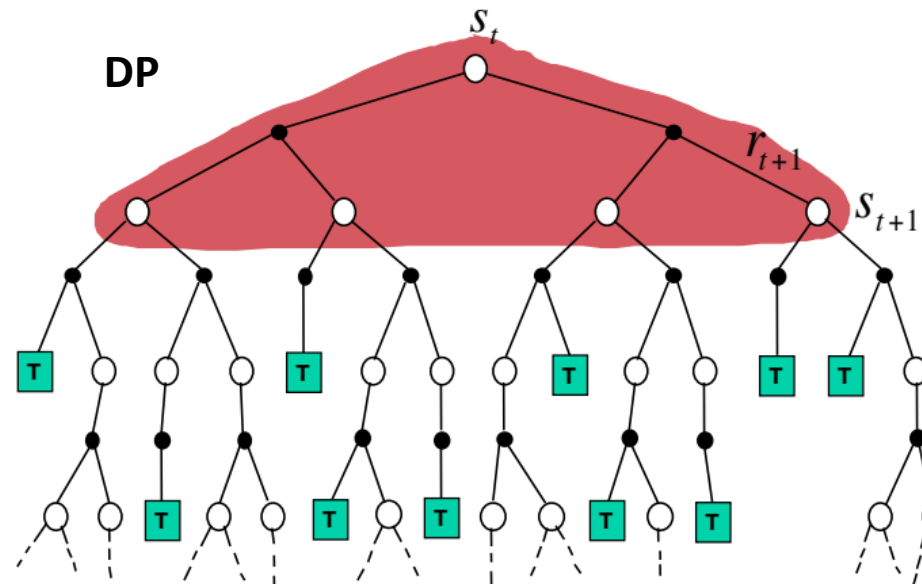
$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

TD



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

DP



Policy Improvement

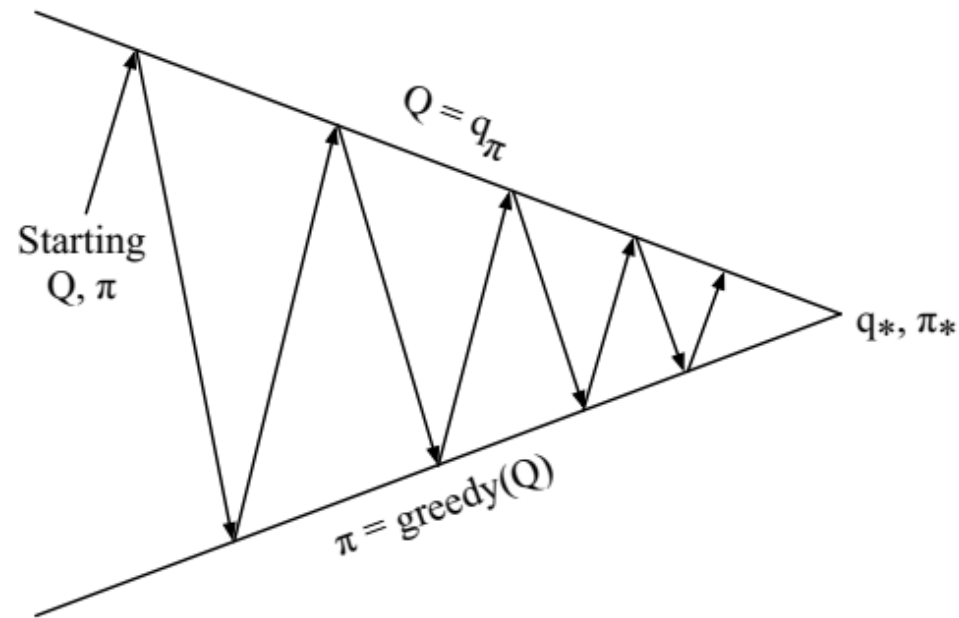
- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

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

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

Policy Iteration



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

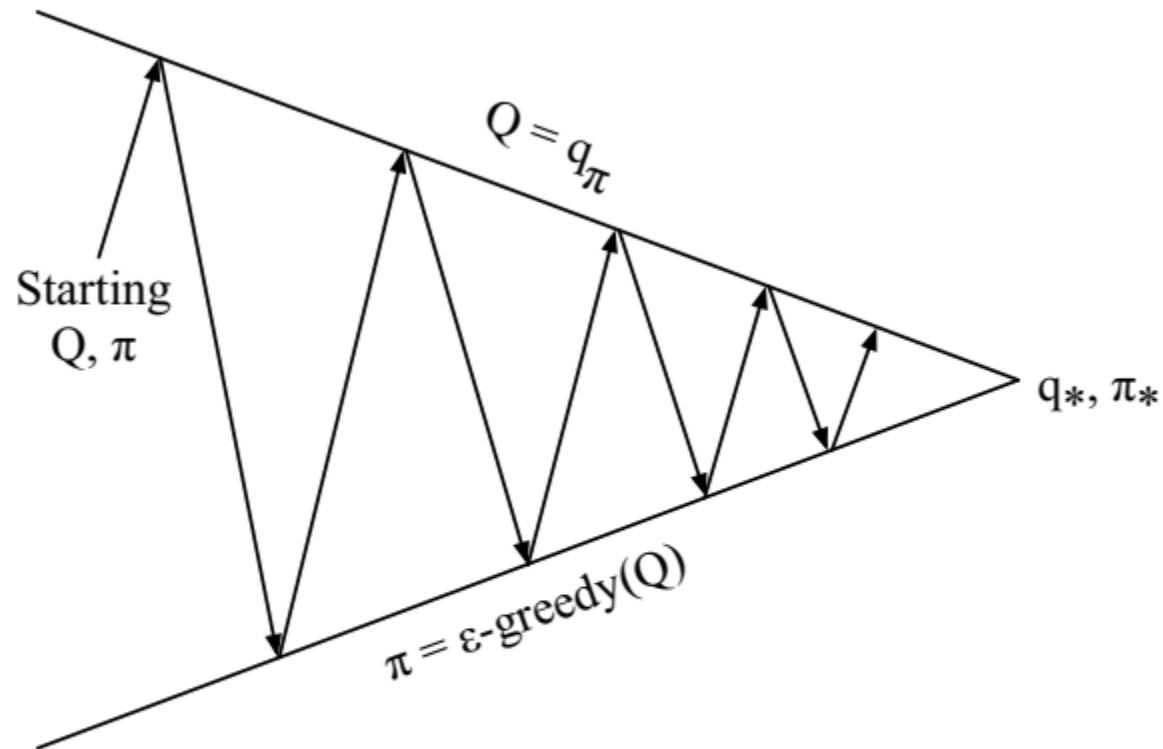
Policy improvement Greedy policy improvement?

ϵ -greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

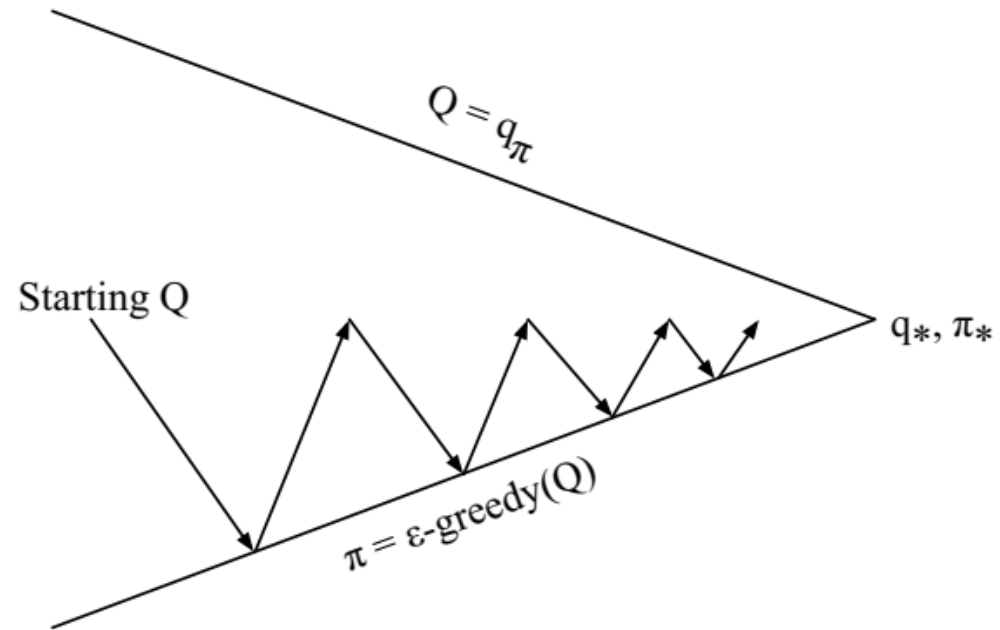
Monte-Carlo Policy Iteration



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

Policy improvement ϵ -greedy policy improvement

Monte-Carlo Control



Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ -greedy policy improvement

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 |
| | 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 store 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

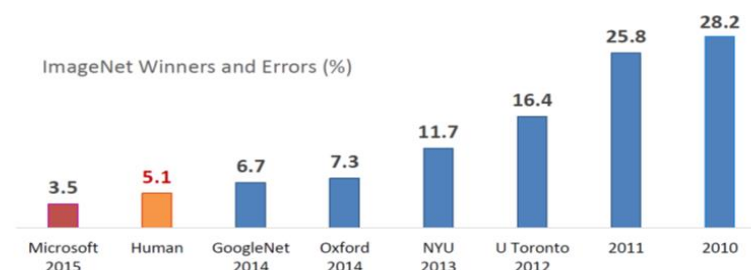
Deep Reinforcement Learning

Deep learning . Value based . Policy gradients

Actor-critic . Model based

Deep Learning Is Making Break-through!

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

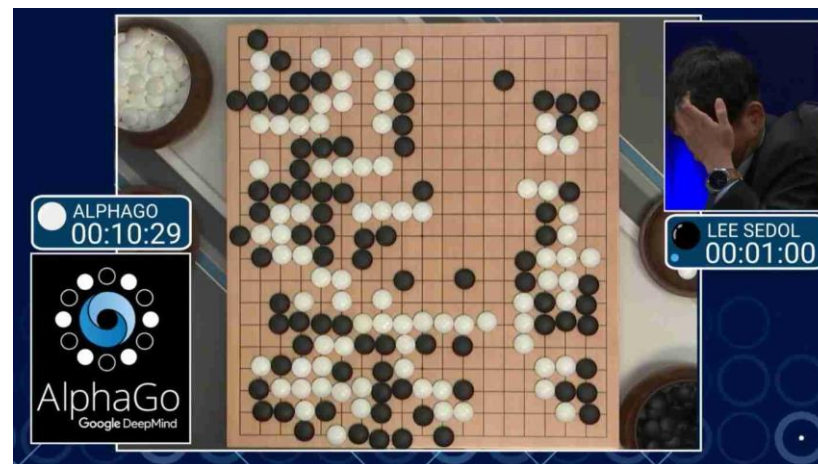
原创 2018-03-15 camel AI科技评论

翻译没有唯一标准答案，它更像是一种艺术。

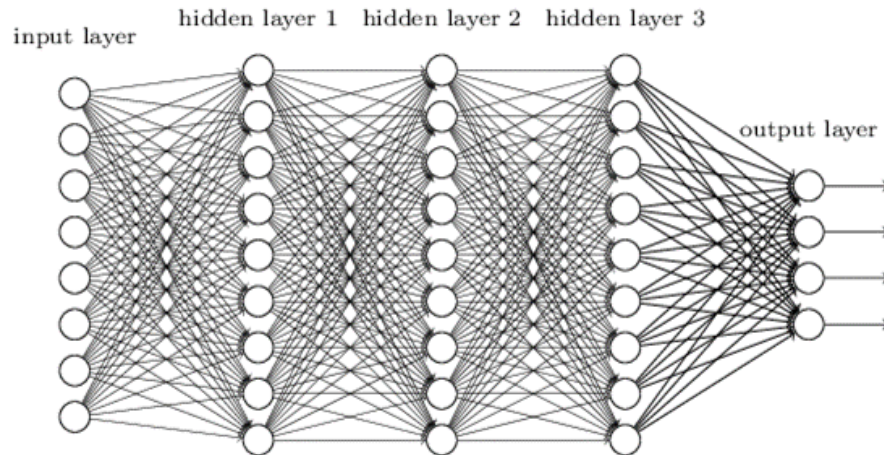
AI科技评论消息：14日晚，微软亚洲研究院与雷德蒙研究院的研究人员宣布，其研发的机器翻译系统在通用新闻报道测试集 newstest2017 的中-英测试集上，达到了可与人工翻译媲美的水平；这是首个在新闻报道的翻译质量和准确率上可以比肩人工翻译的翻译系统。



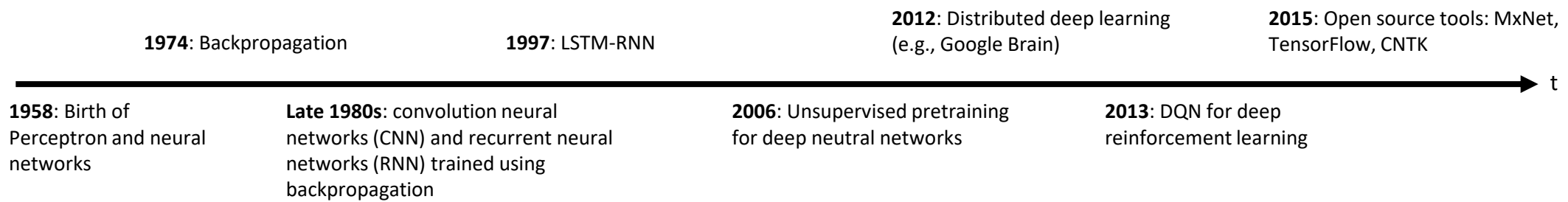
2016年10月，微软的语音识别系统在日常对话数据上，达到了5.9%的单词错误率，首次取得与人类相当的识别精度



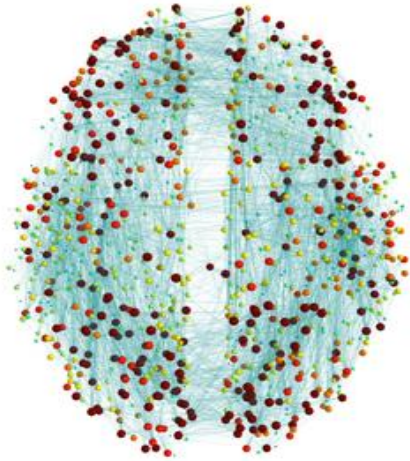
Deep Learning



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](#).



Driving Power



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



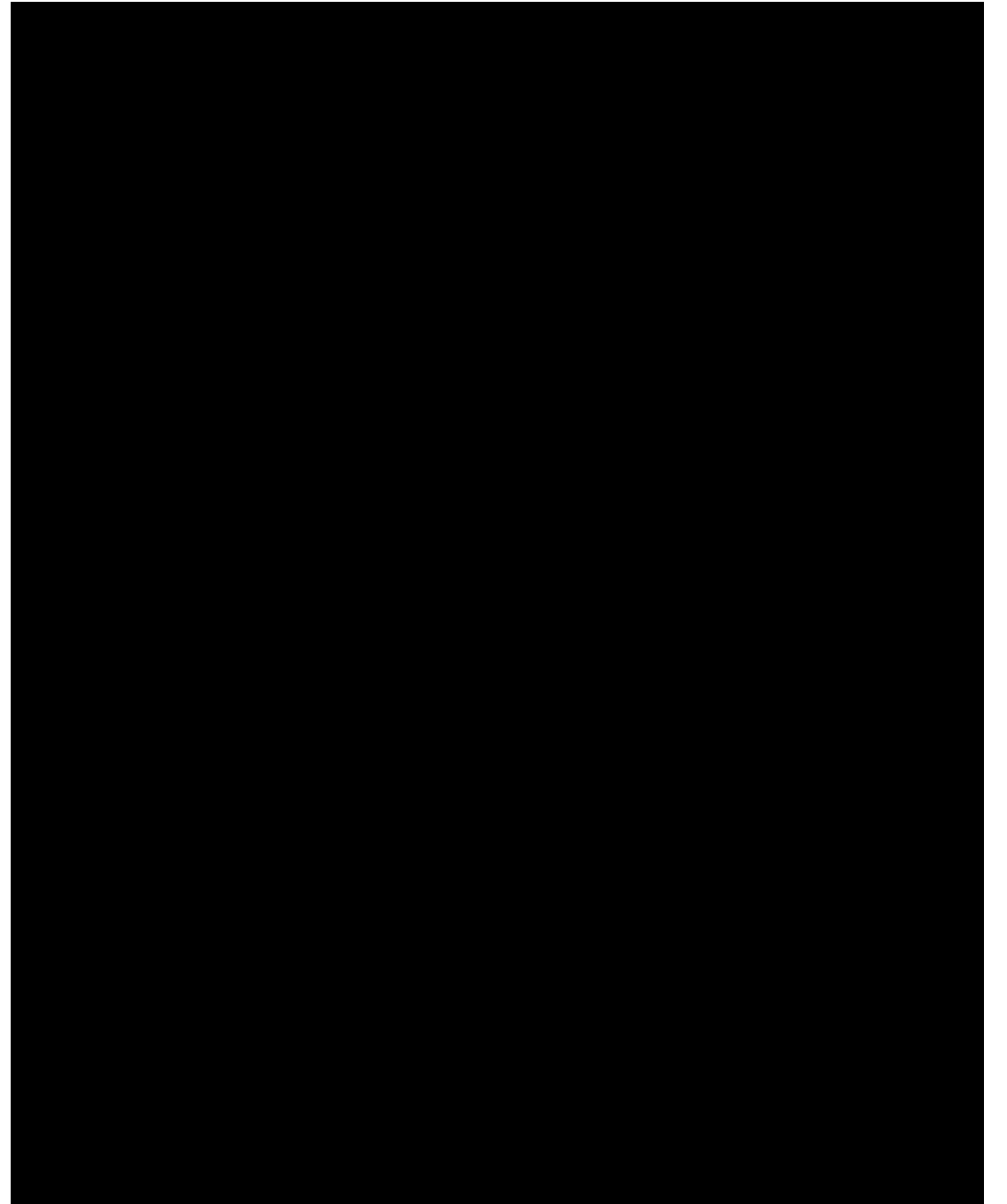
- **Big computer clusters:** CPU clusters, GPU clusters, FPGA farms, provided by Amazon, Azure, etc.



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

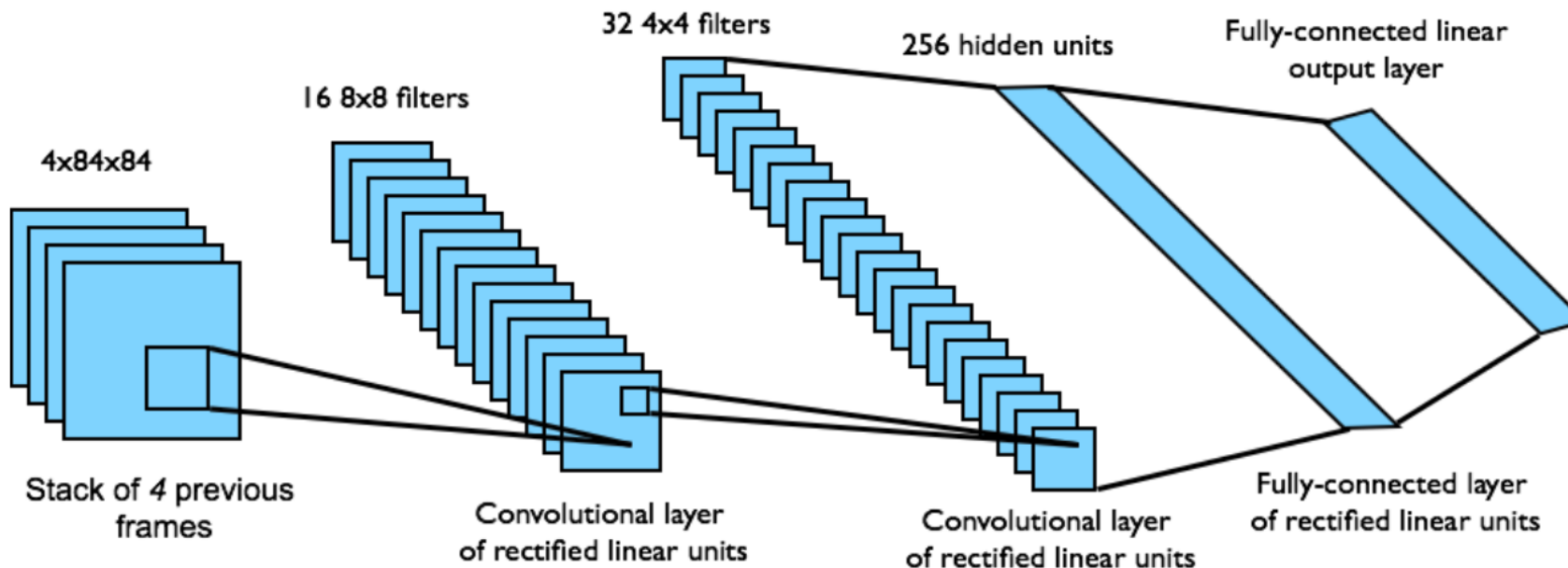
Value based methods: estimate value function or Q-function of the optimal policy (no explicit policy)

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 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

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

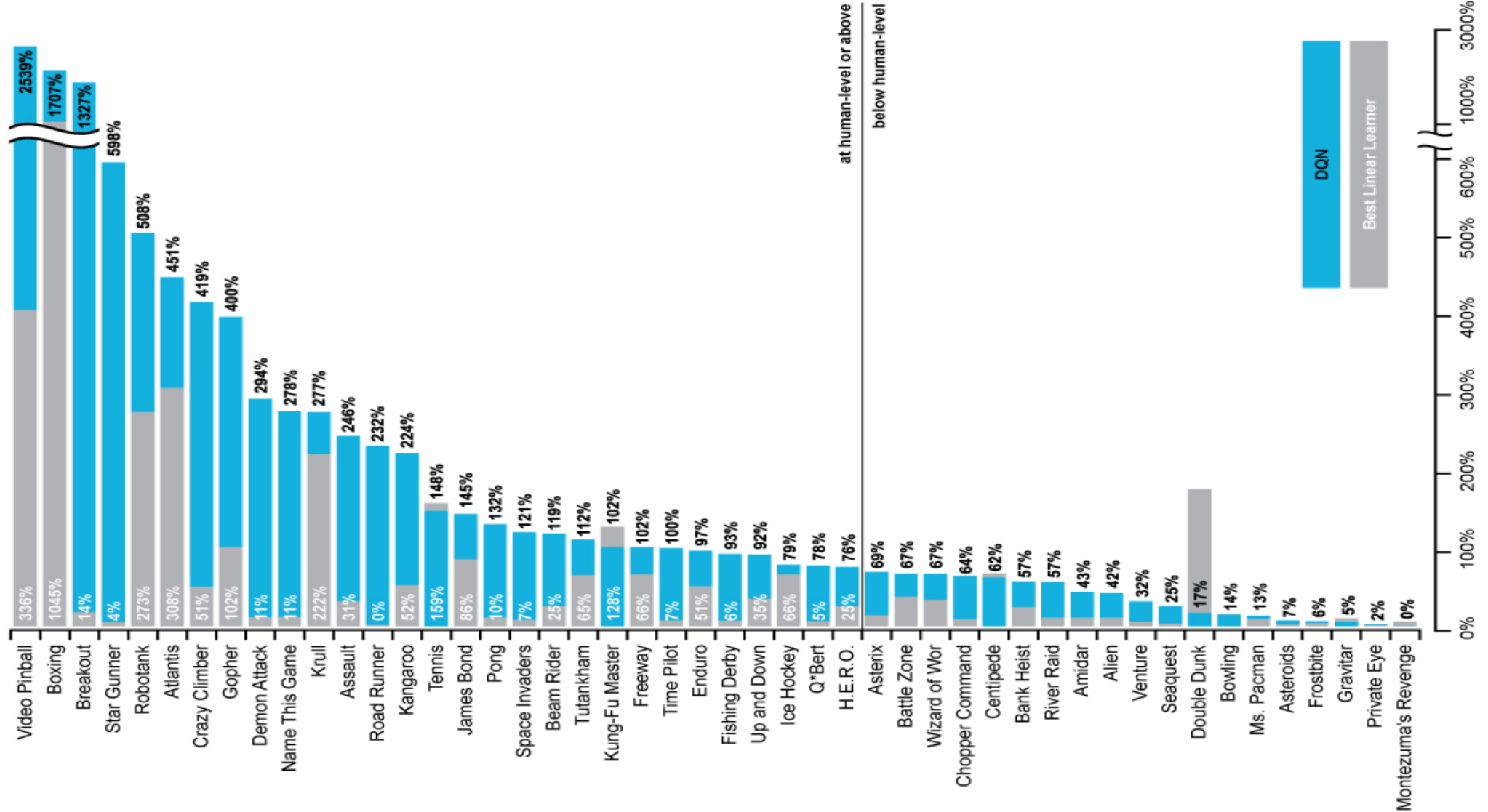
- Periodically update fixed parameters $\theta^- \leftarrow \theta$

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 |

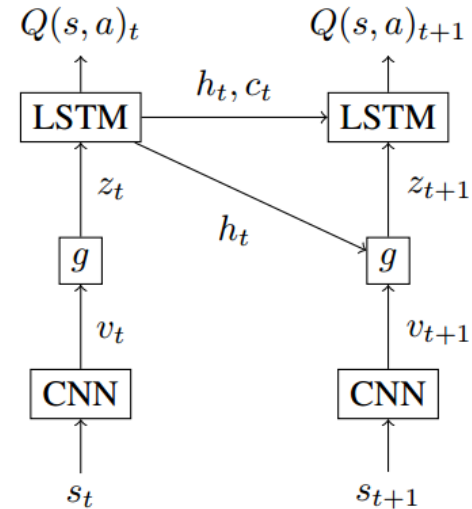
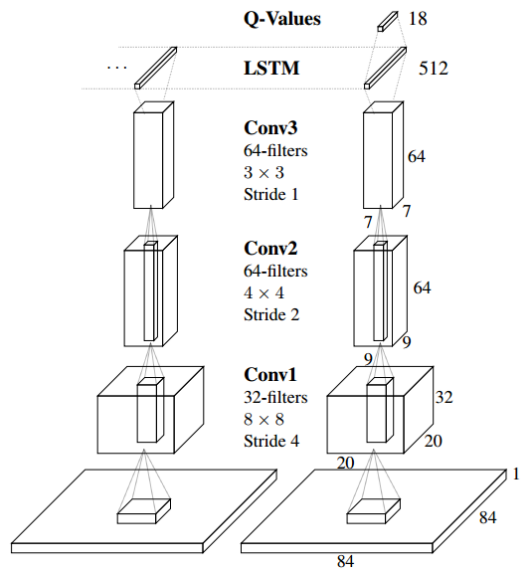
Other Tricks

- 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

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

a convenient identity

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

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

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

Policy Gradients

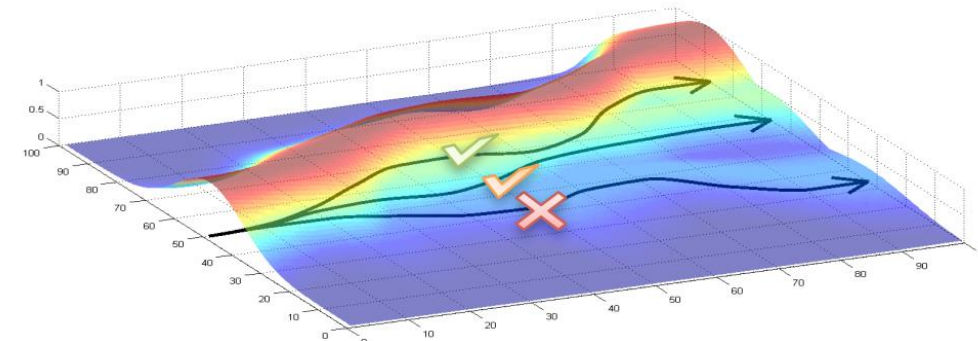
- Optimization Problem: Find θ that maximizes expected total reward.
 - The gradient of a stochastic policy $\pi_\theta(a|s)$ is given by

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

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

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

- 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.
 - $v_t = r_1 + r_2 + \dots + r_t$

function REINFORCE

Initialise θ arbitrarily

for each episode $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **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 $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\}$

for $t = 1$ to $T - 1$ **do**

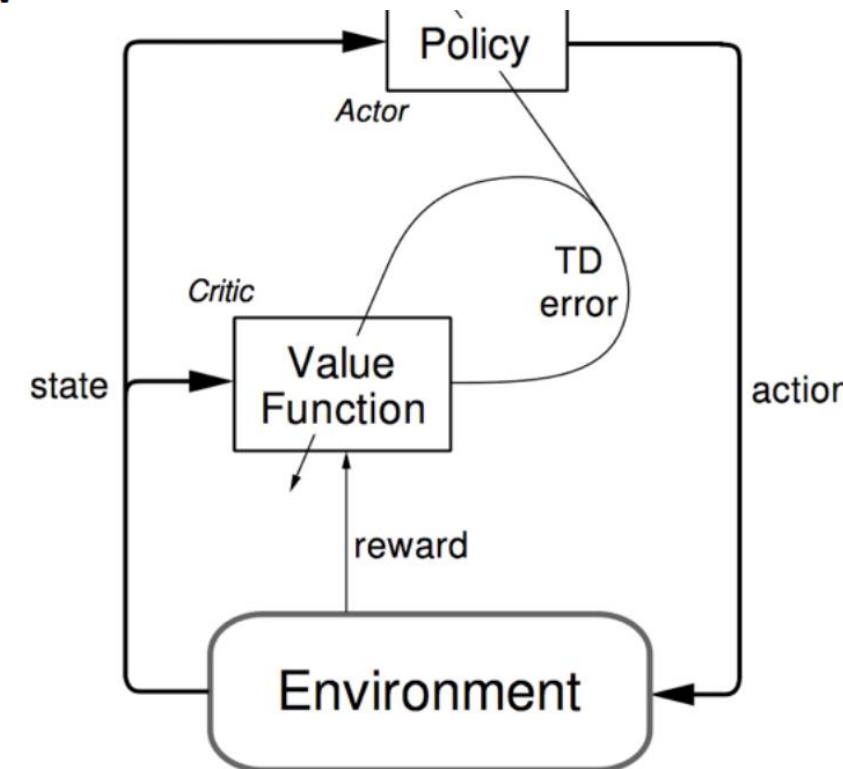
$\theta \leftarrow \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) v_t$

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

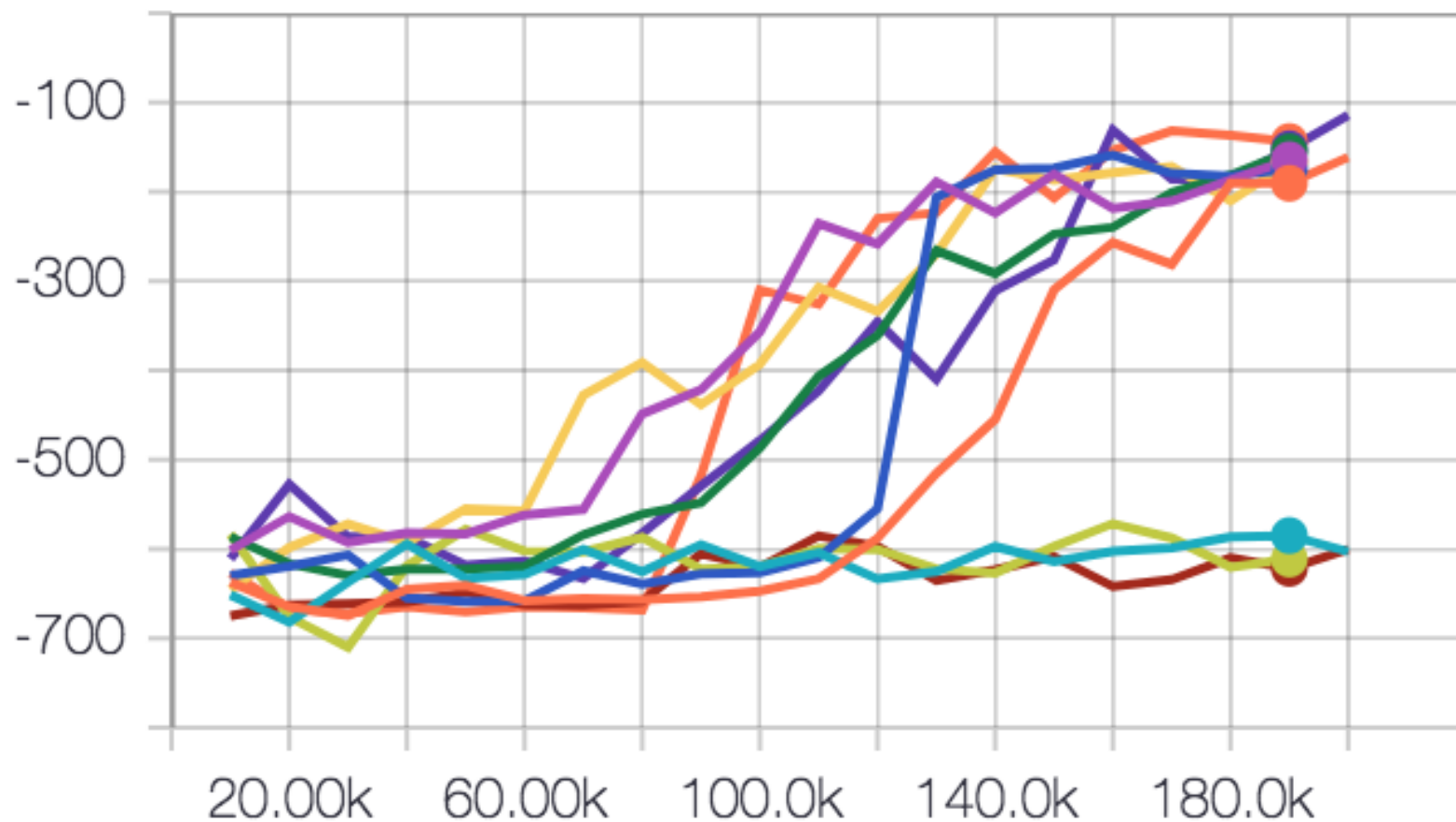
Model based DRL: Challenges

- 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

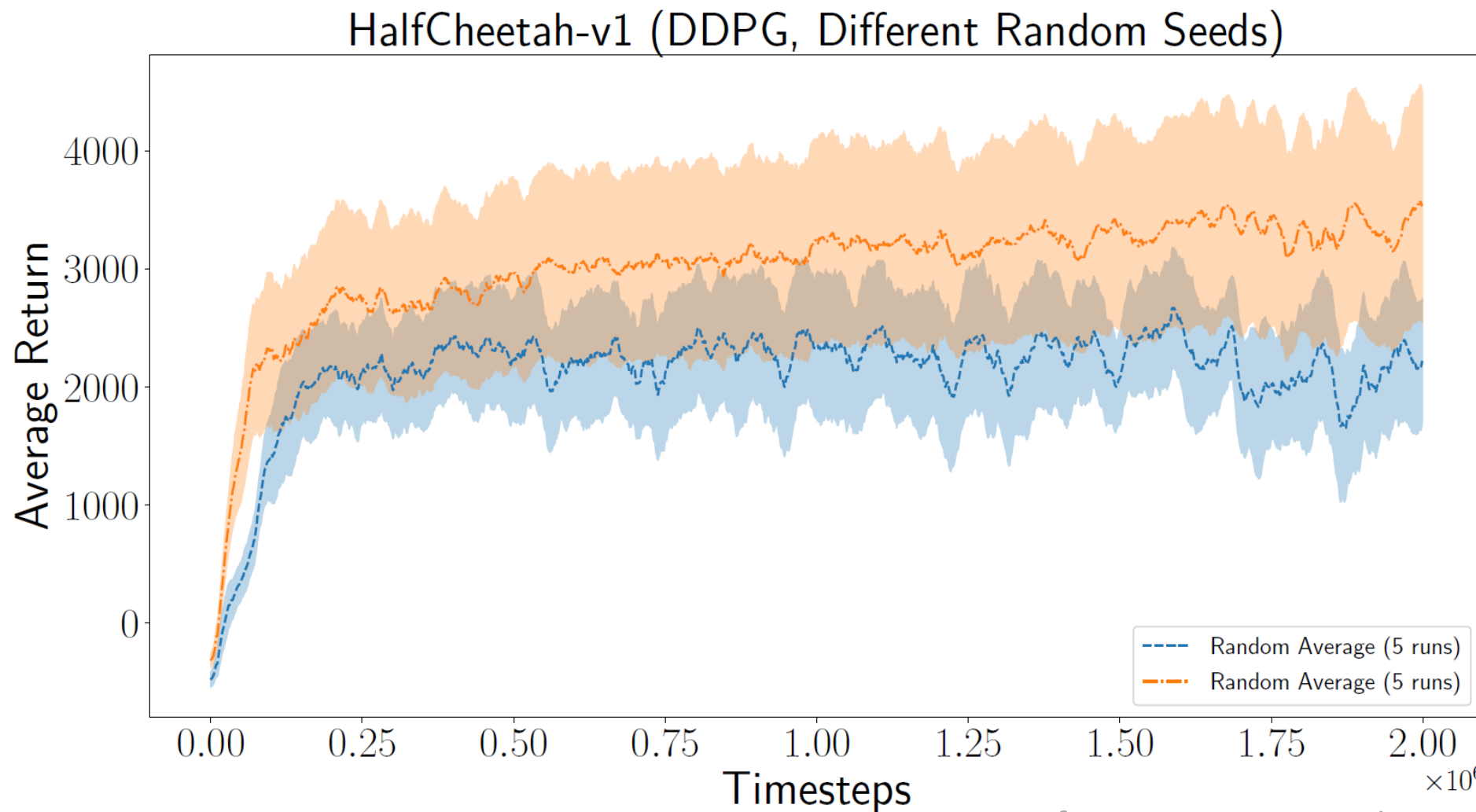
Challenges and Opportunities

1. Robustness – random seeds

episode_reward/test

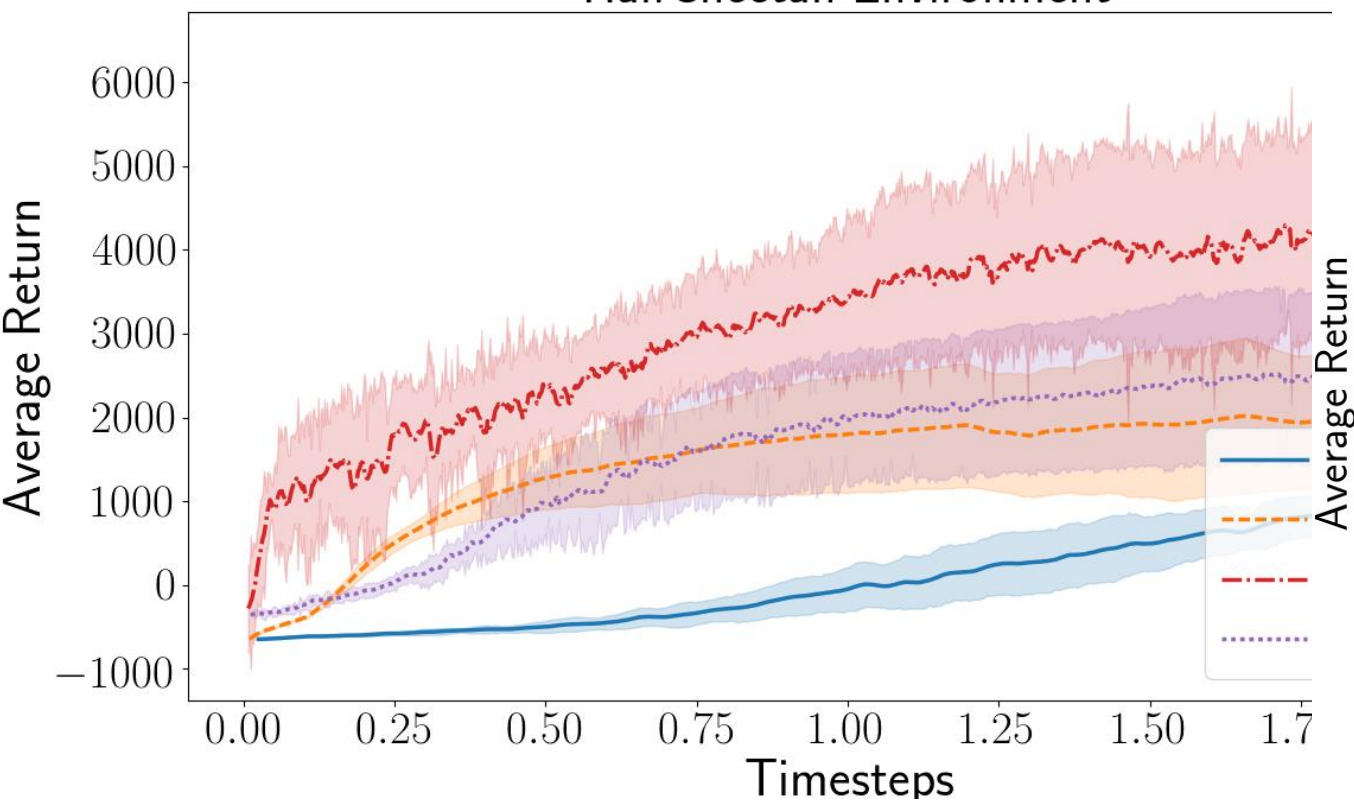


1. Robustness – random seeds

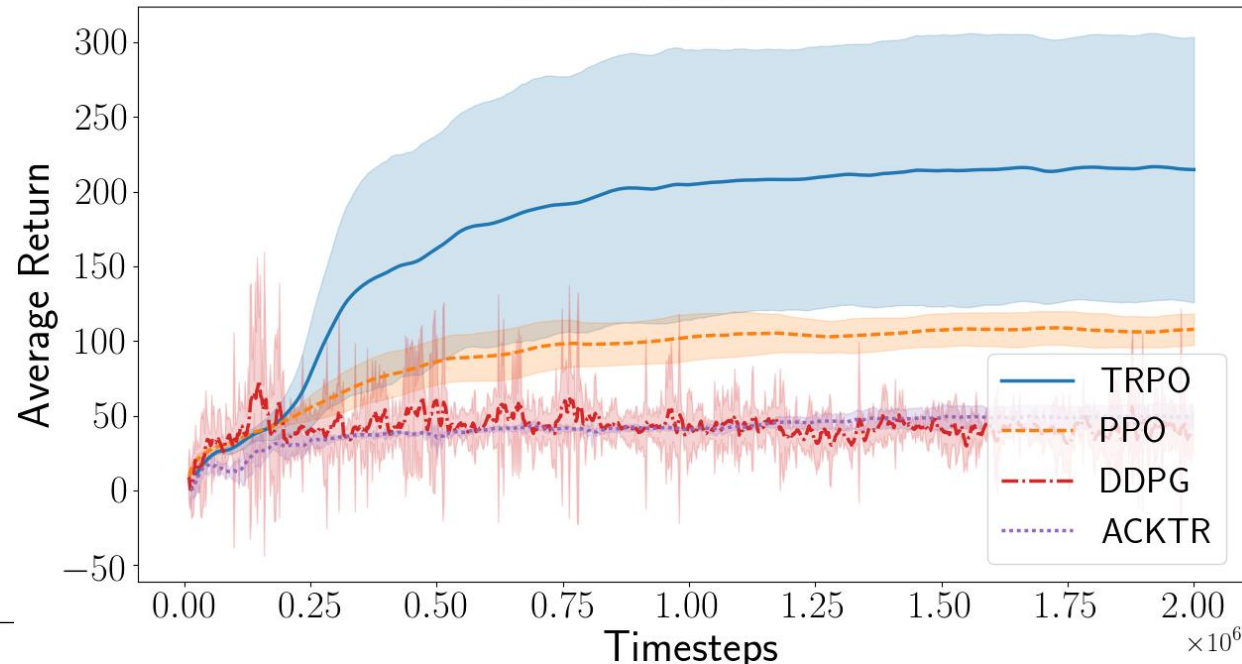


2. Robustness – across task

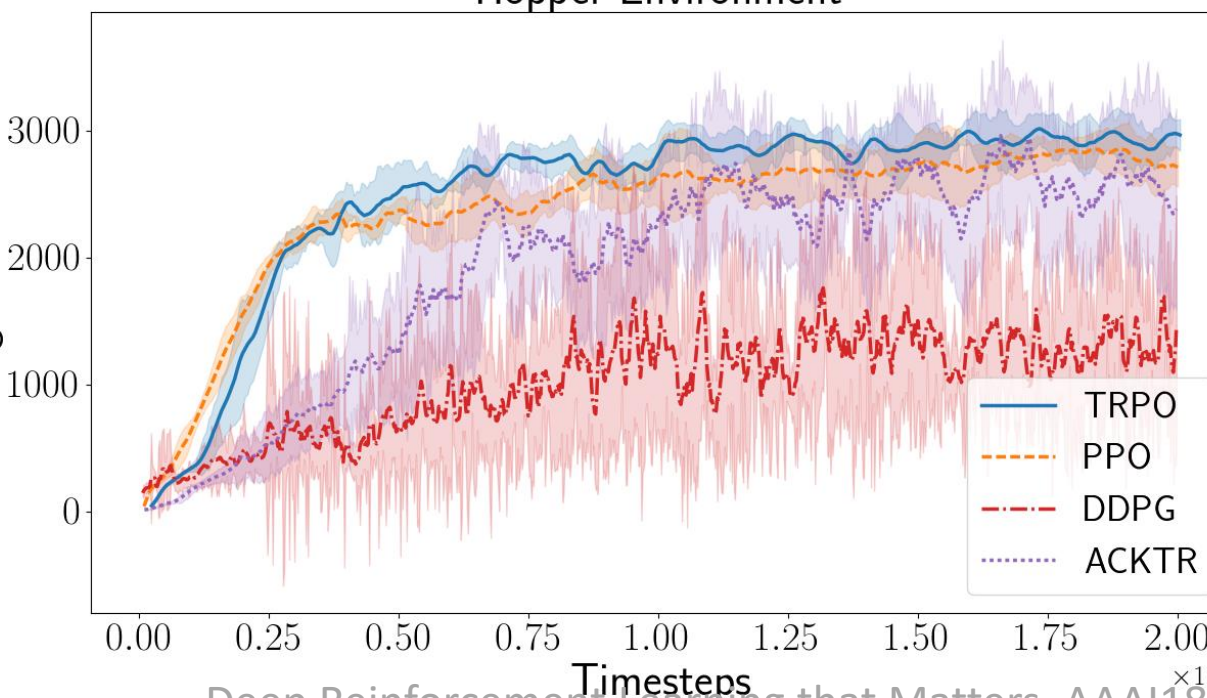
HalfCheetah Environment



Swimmer Environment



Hopper Environment

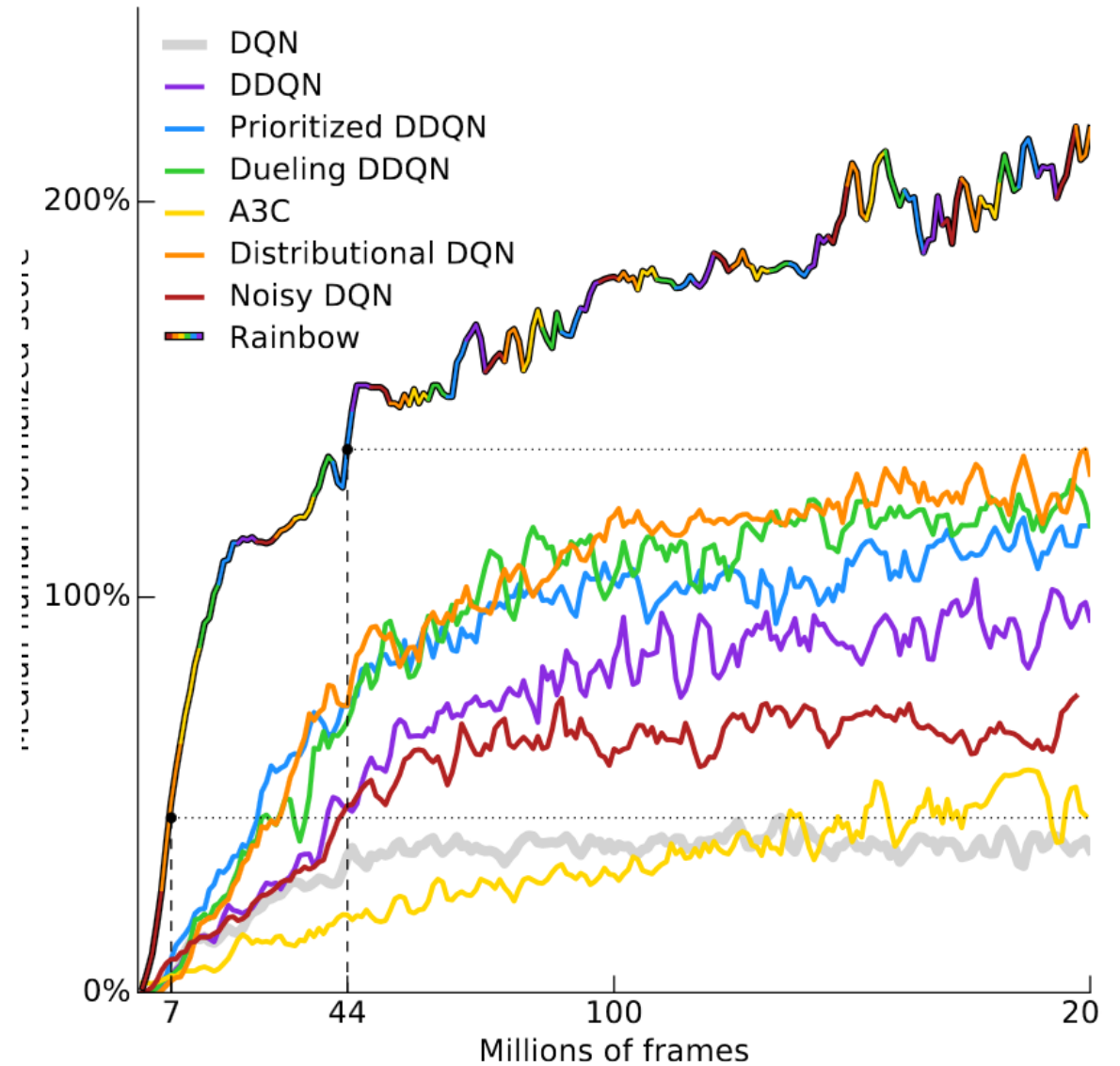


As a Comparison

- 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

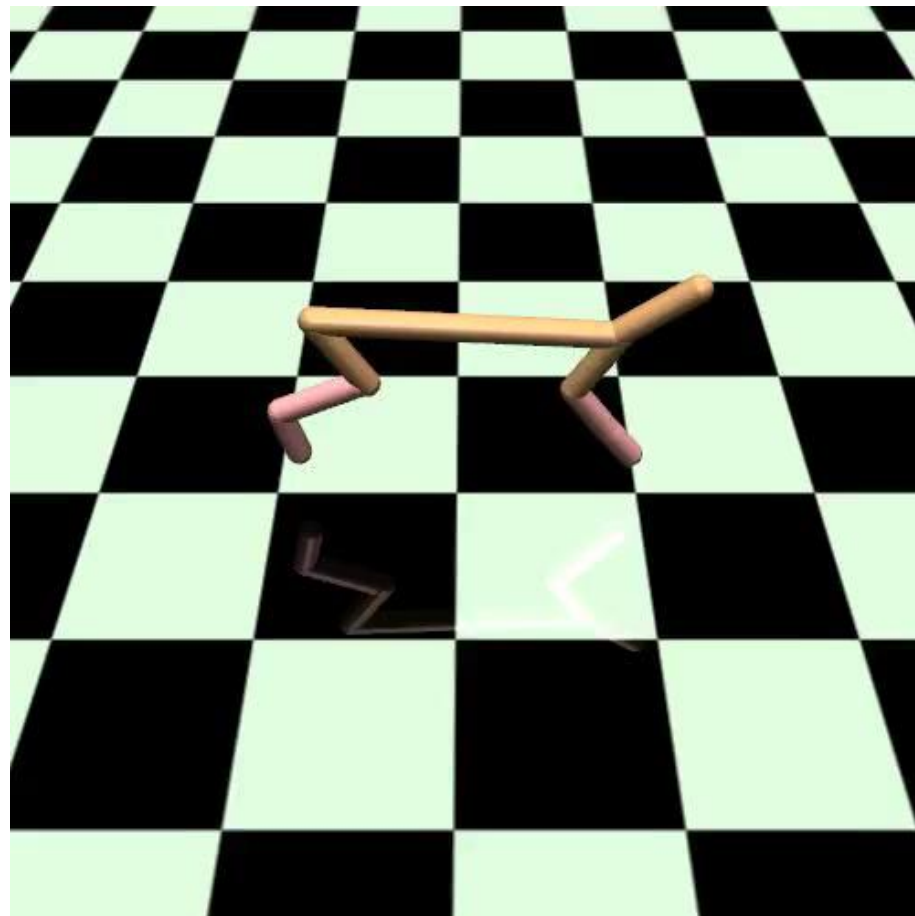


Rainbow: Combining Improvements in Deep Reinforcement Learning

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:

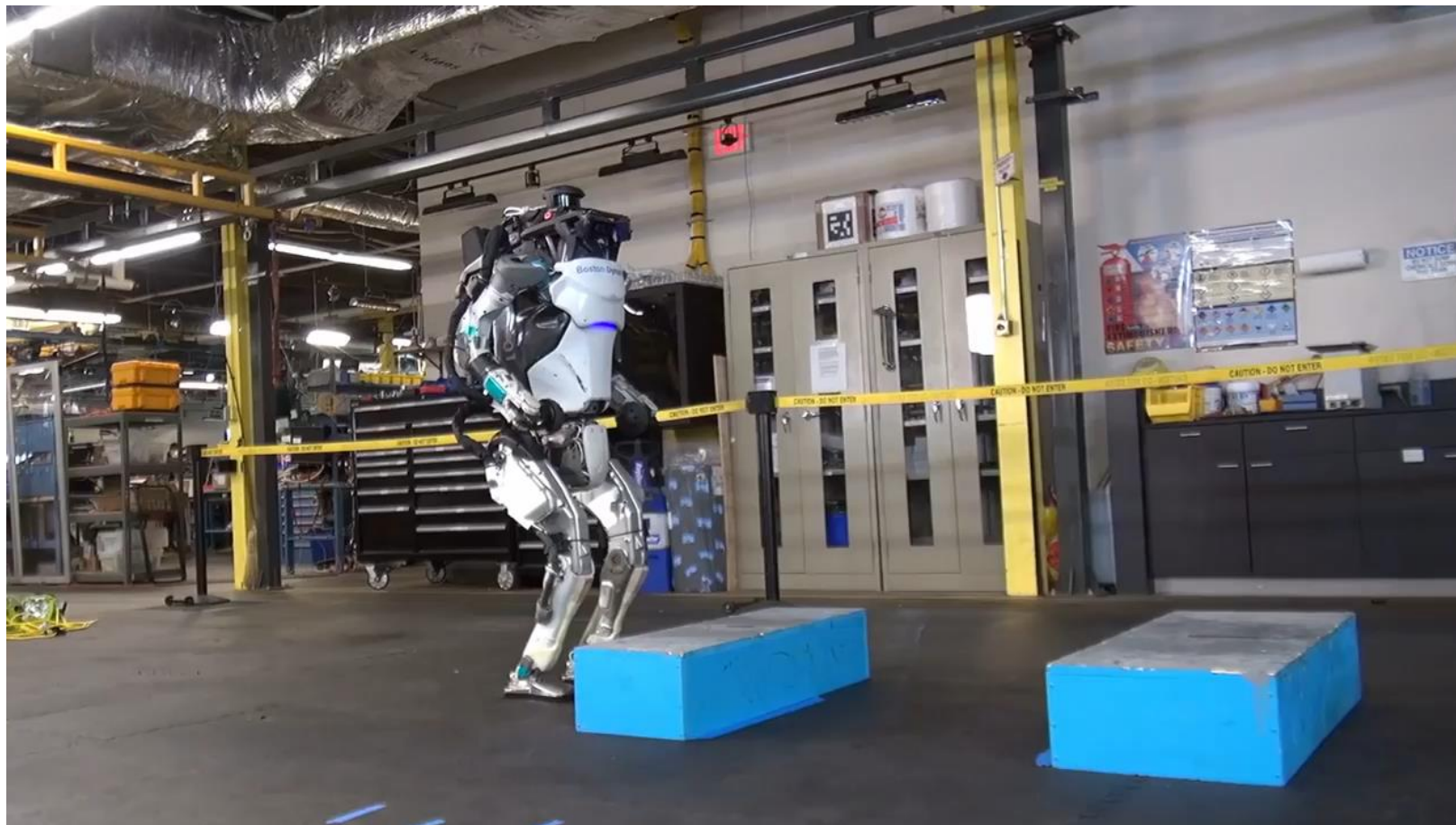
- Usually no (visible) immediate reward for each action
- Maybe no (visible) explicit final reward for a sequence of actions
- Don't know how to terminate a sequence

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, Houthoofd 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

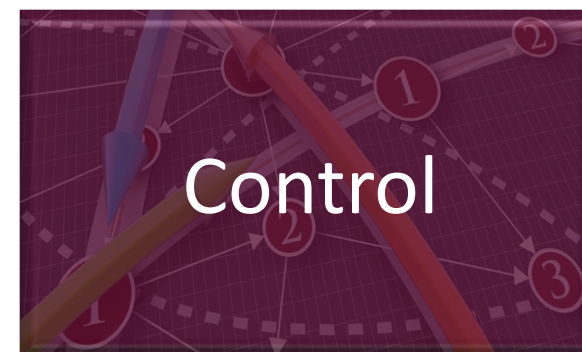
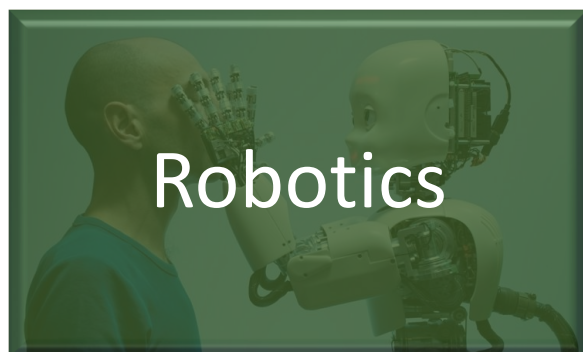
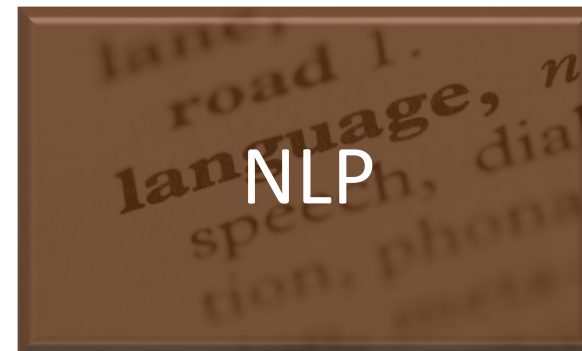
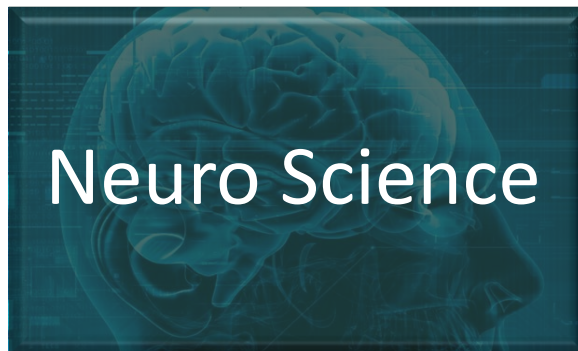
Better optimization

Define reward in practical applications

Identify appropriate tasks

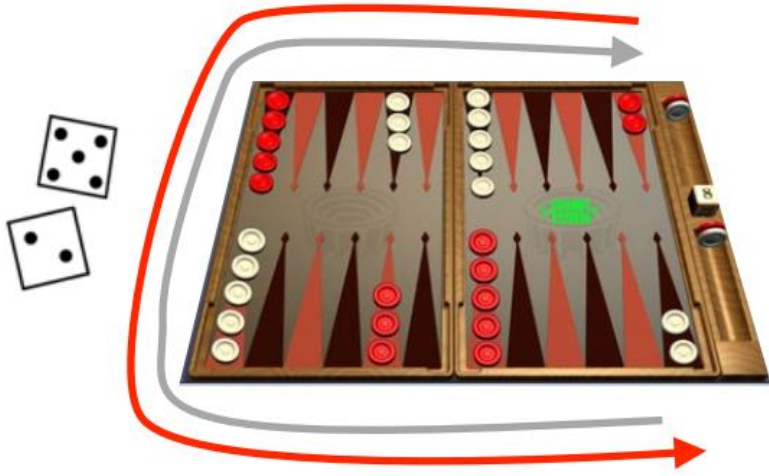
Imperfect information and multi-agent games

Applications



Game

- RL for Game
 - Sequential Decision Making
 - Delayed Reward



TD-Gammon



Atari Games

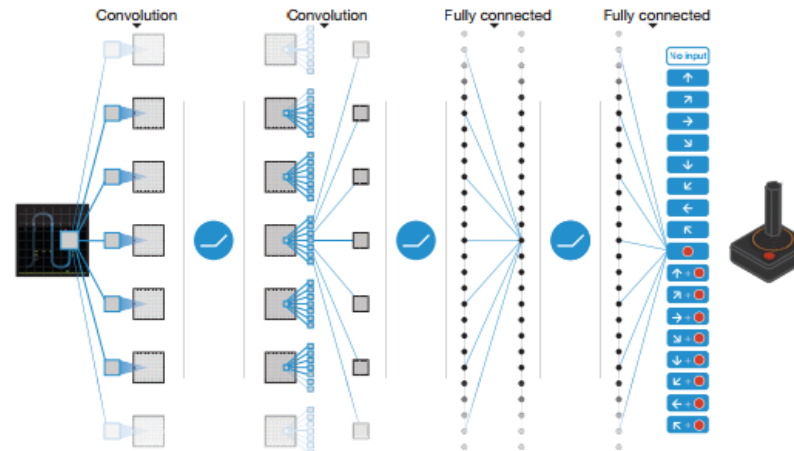
Game



- 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

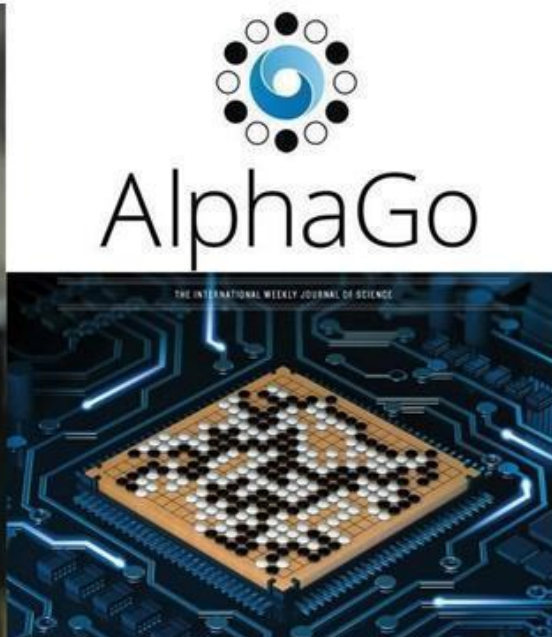
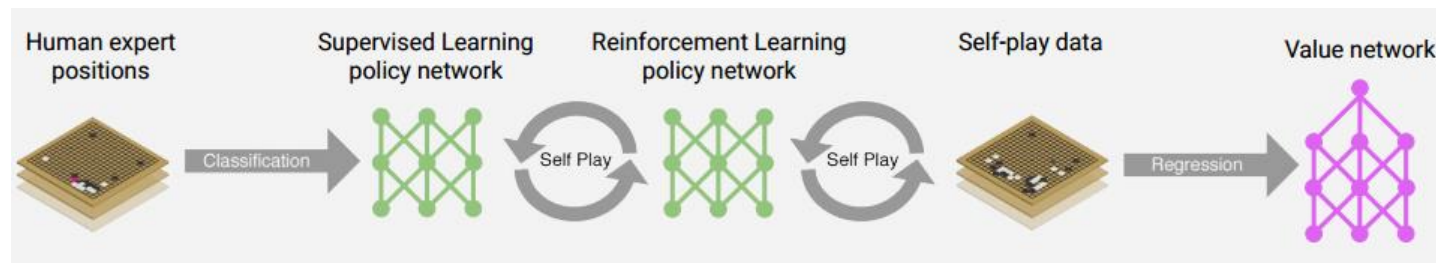
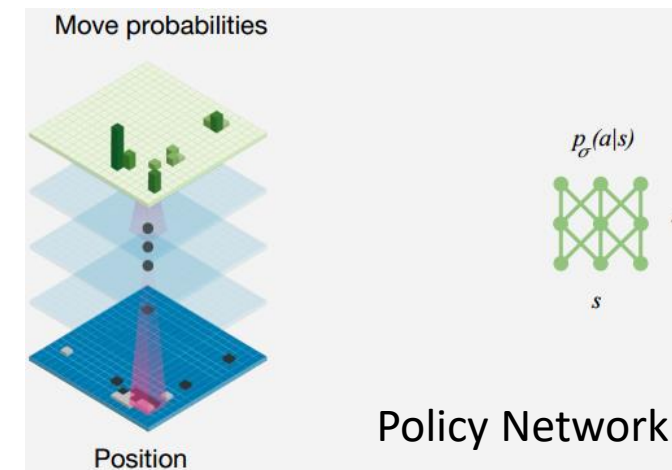
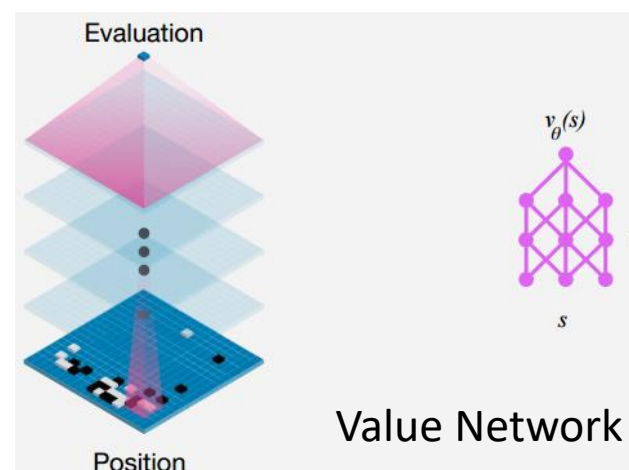
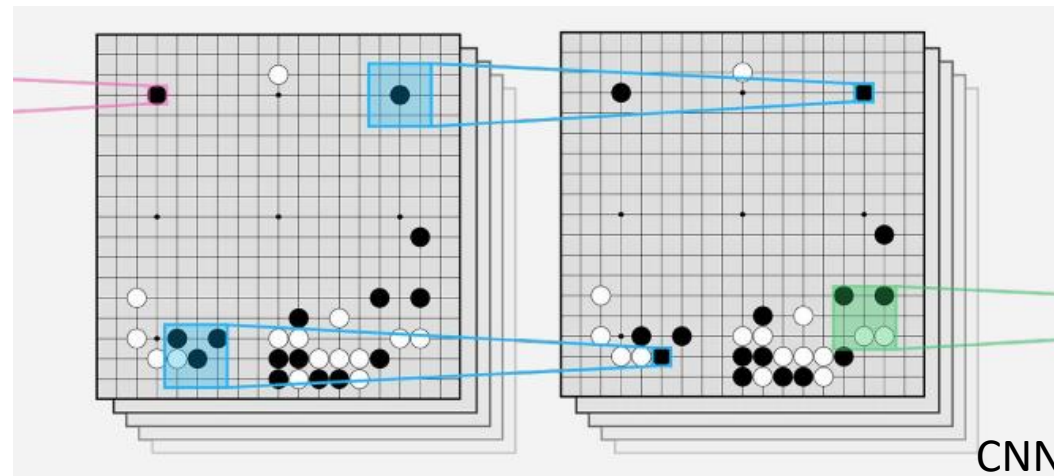
mapping raw
screen pixels

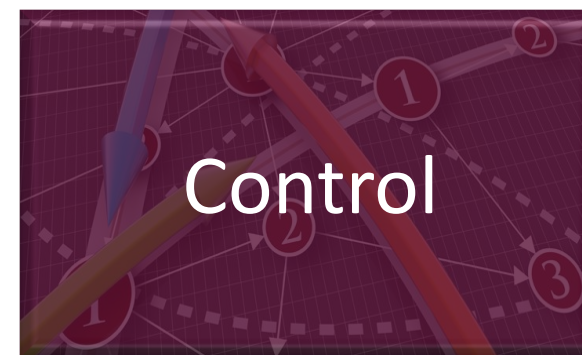
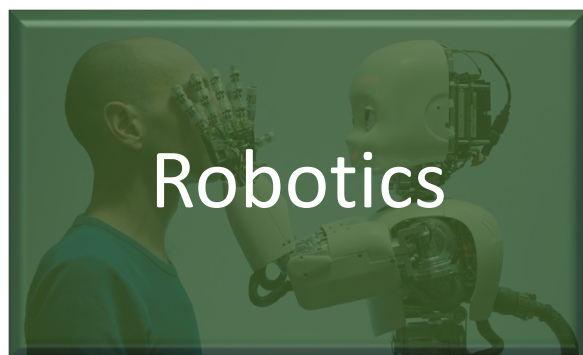
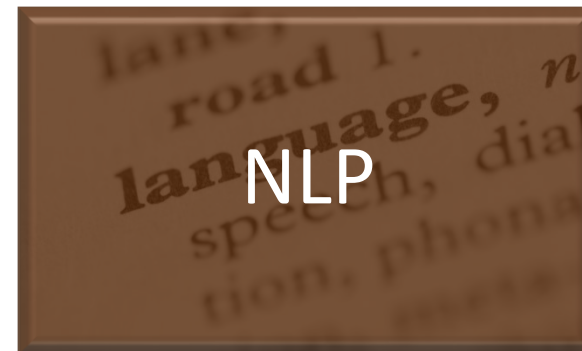
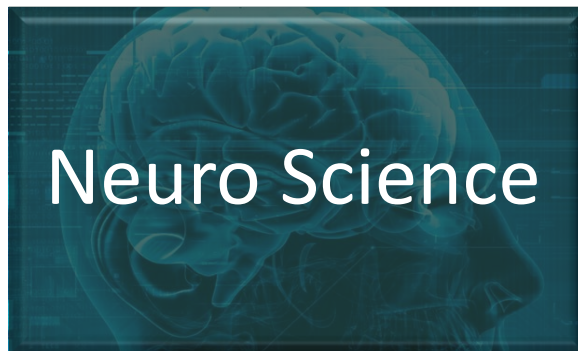


to predictions
of final score
for each of 18
joystick actions

Game

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





Neuro Science

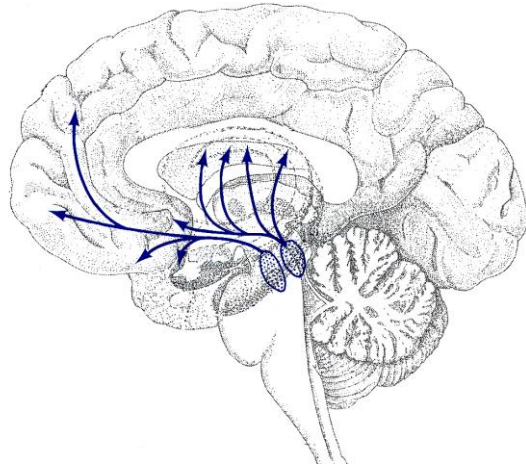


**The world presents animals/humans with a huge reinforcement learning problem
(or many such small problems)**

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

What is dopamine?



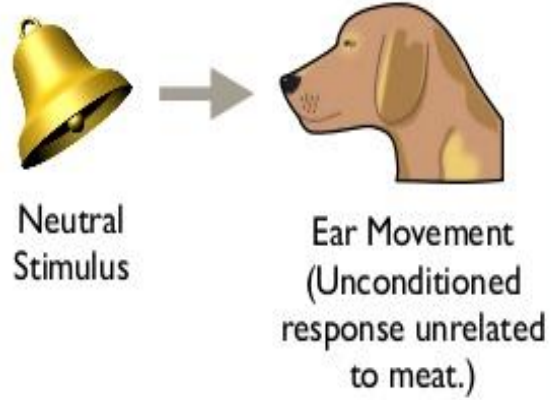
Dopamine

$C_8H_{11}NO_2$

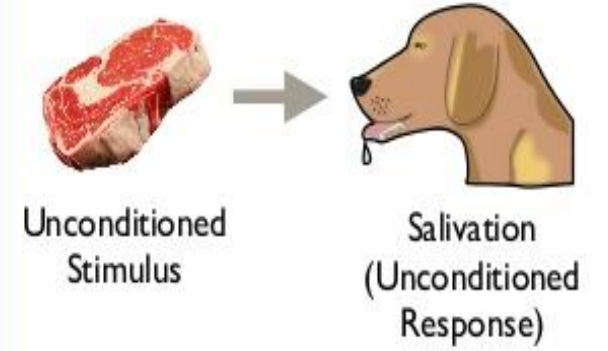




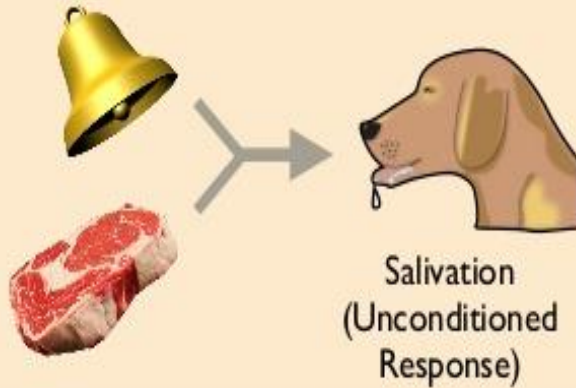
1. Before Conditioning



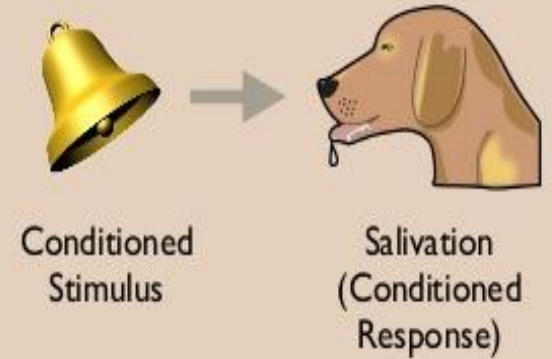
2. Before Conditioning

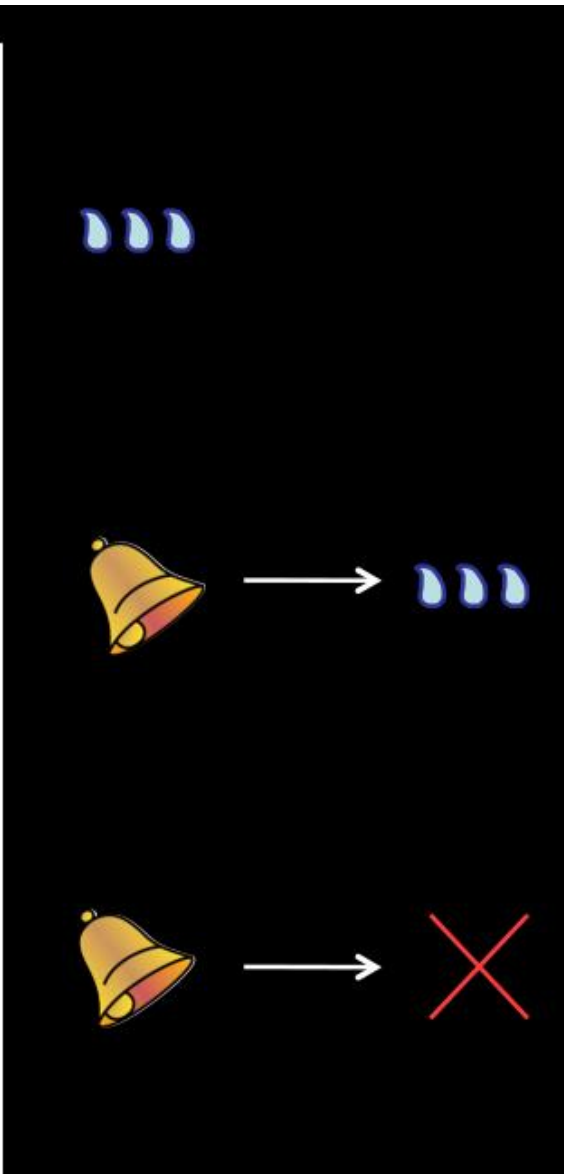
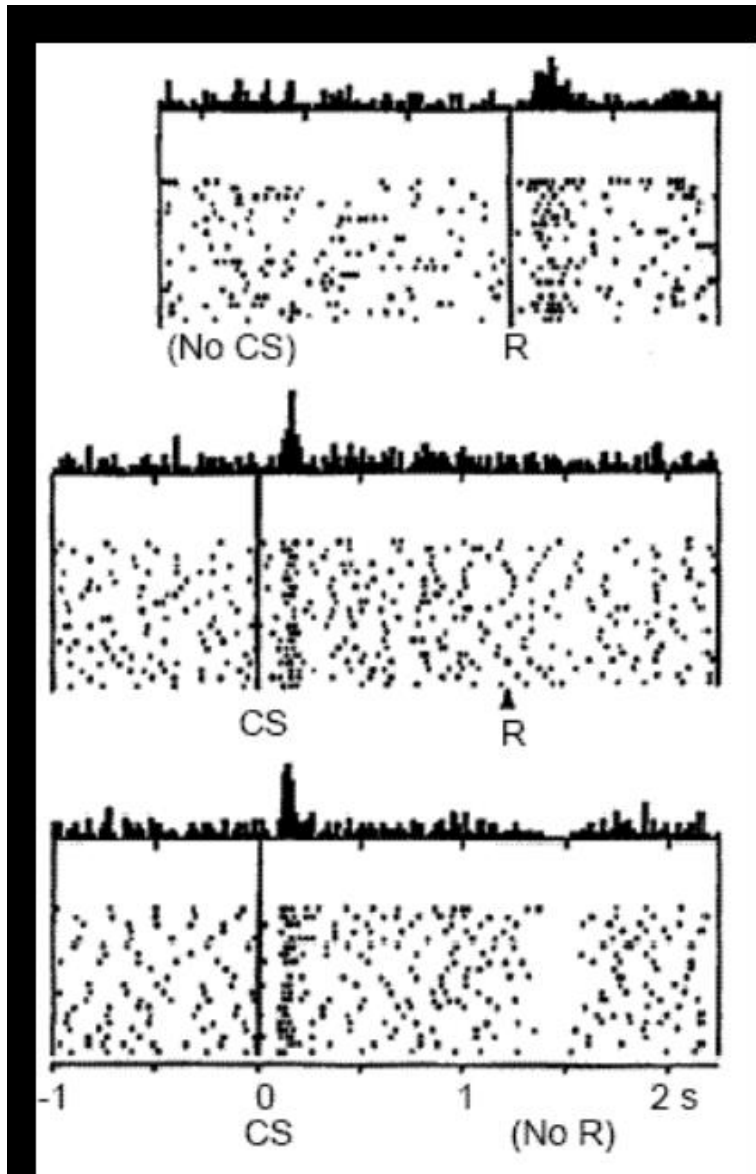


3. During Conditioning



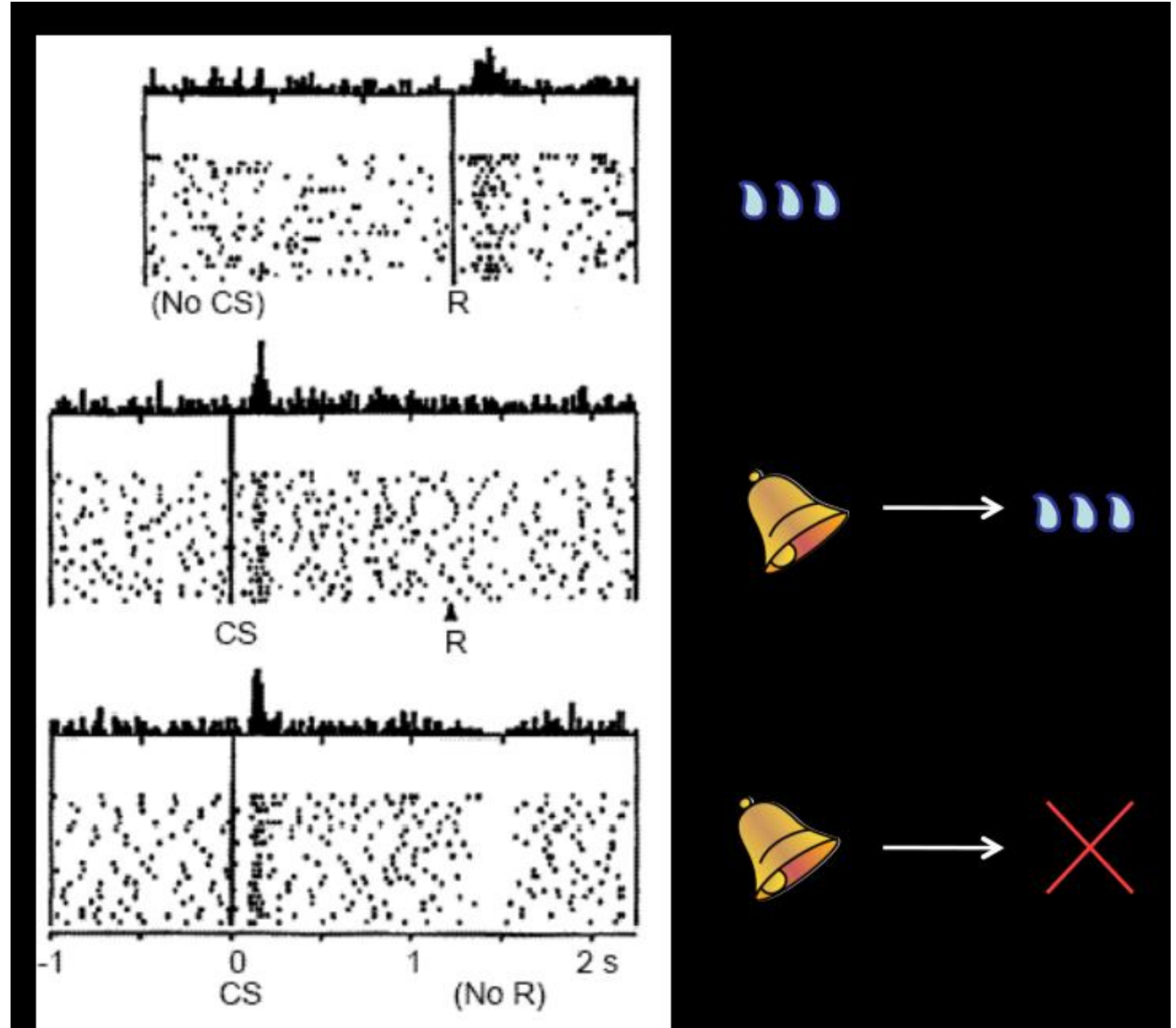
4. After Conditioning





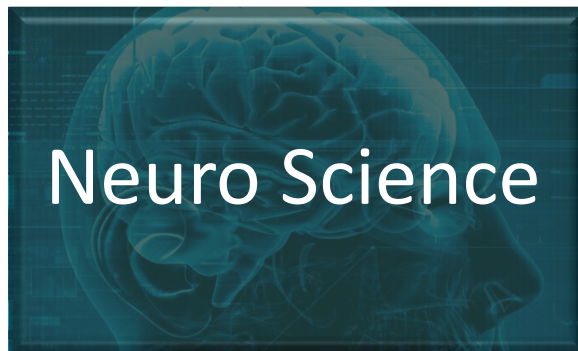
The idea: Dopamine encodes a temporal difference reward prediction error

(Montague, Dayan, Barto mid 90's)





Game



Neuro Science



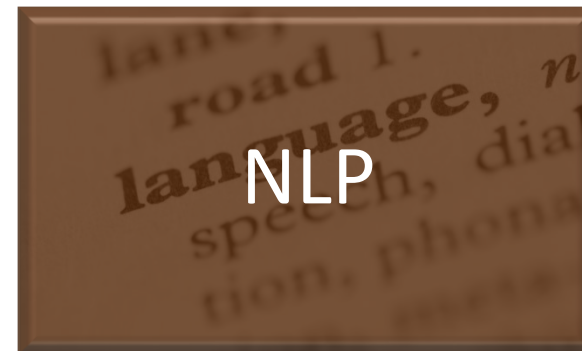
Music & Movie



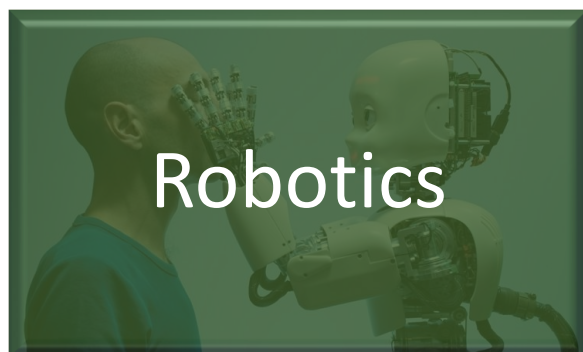
Healthcare



Trading



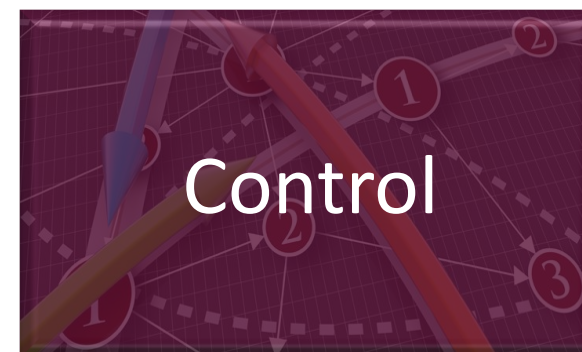
NLP



Robotics



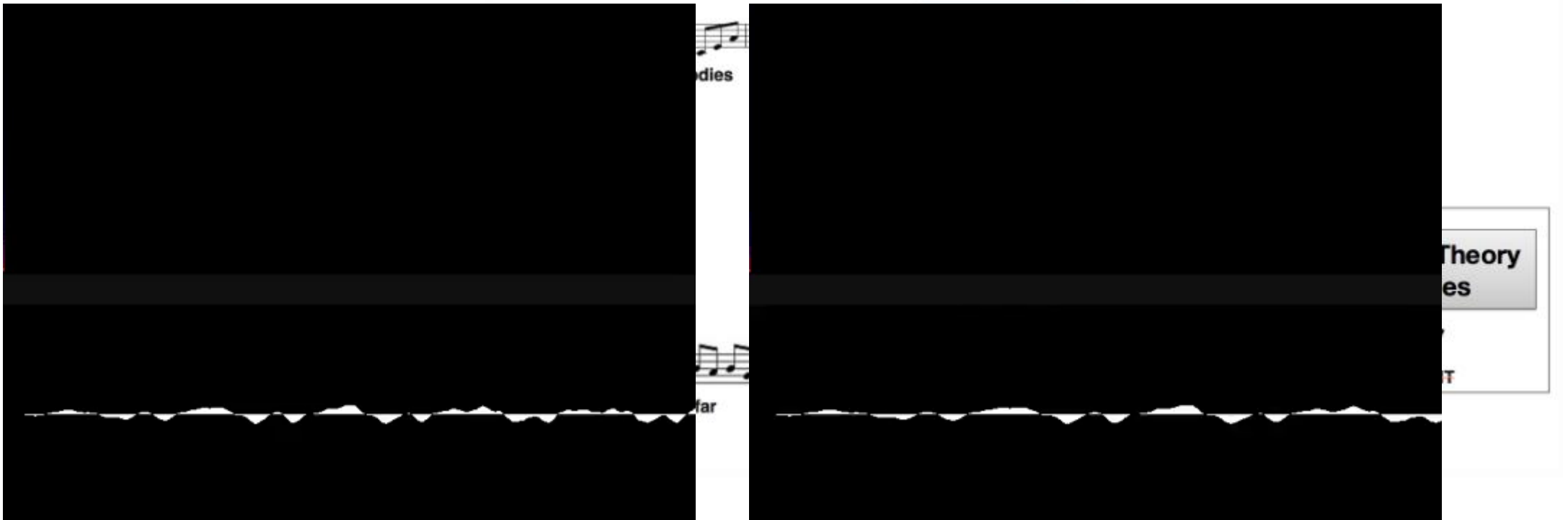
Education



Control

Music & Movie

- Music
 - Tuning Recurrent Neural Networks with Reinforcement Learning
 - [LSTM](#) v.s. [RL tuner](#)



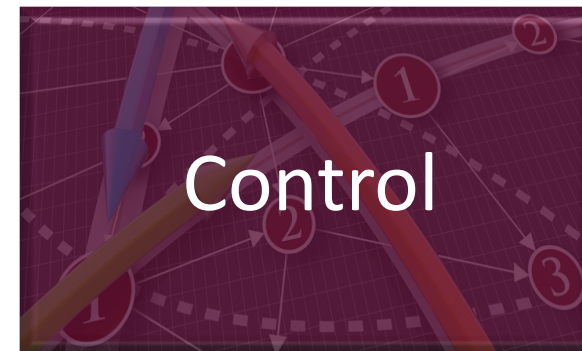
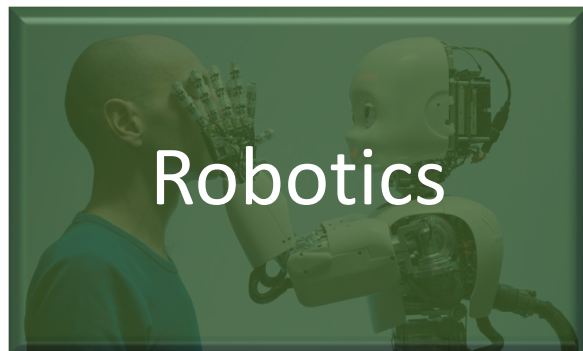
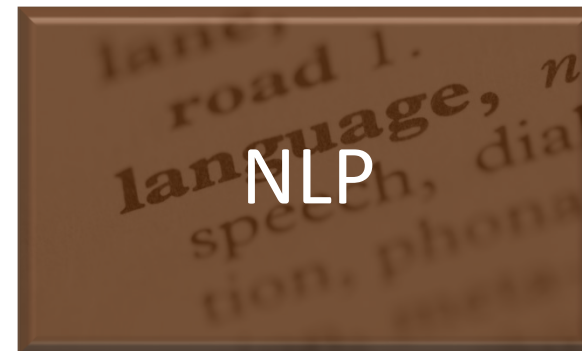
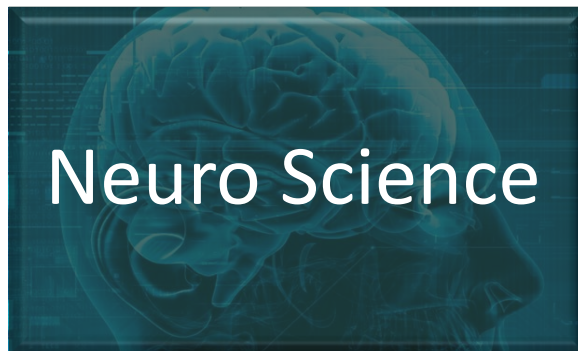
Music & Movie

- Movie

Terrain-Adaptive Locomotion Skills using Deep Reinforcement Learning



Xue Bin Peng, Glen Berseth, Michiel van de Panne
University of British Columbia



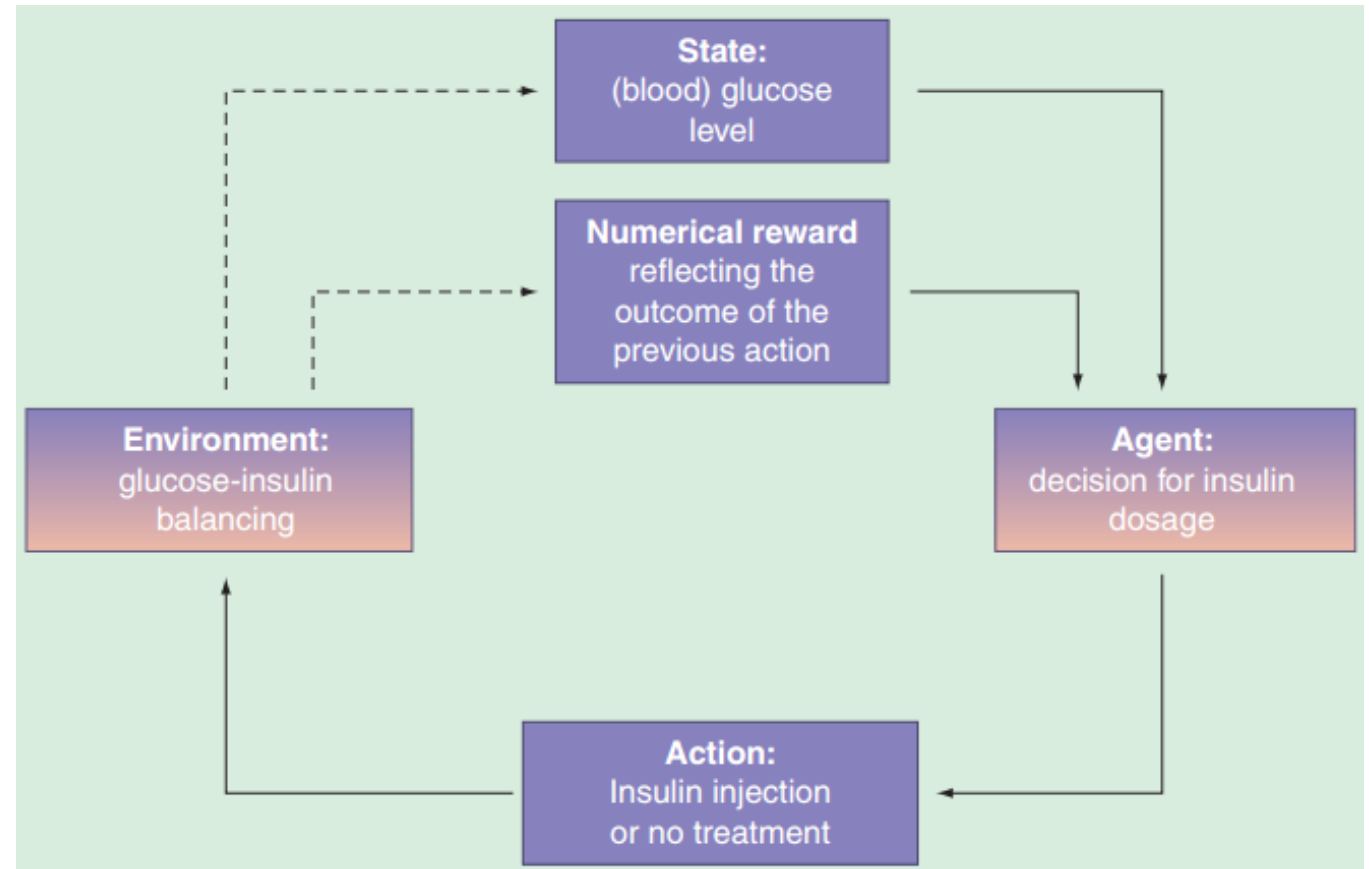
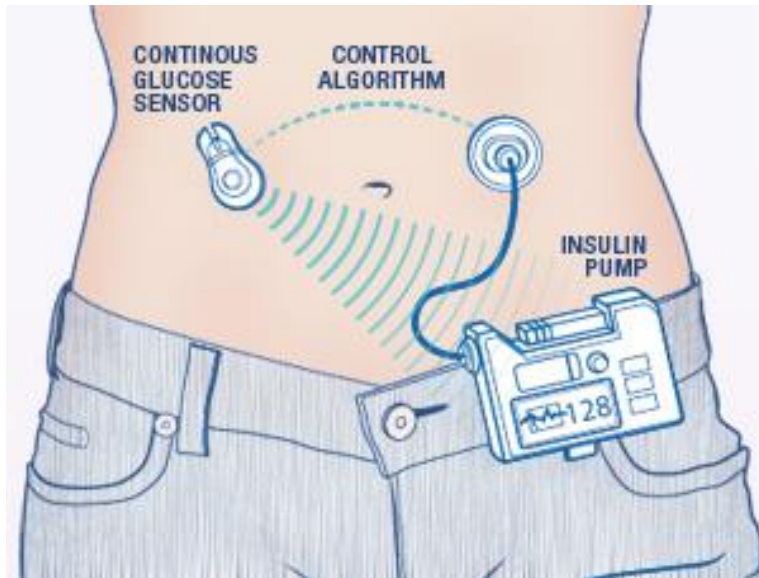
HealthCare

- Sequential Decision Making in HealthCare



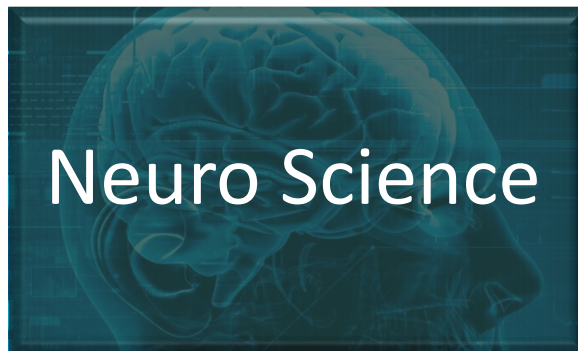
HealthCare

- Artificial Pancreas





Game



Neuro Science



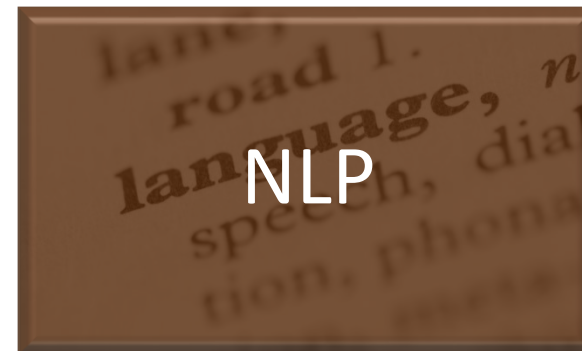
Music & Movie



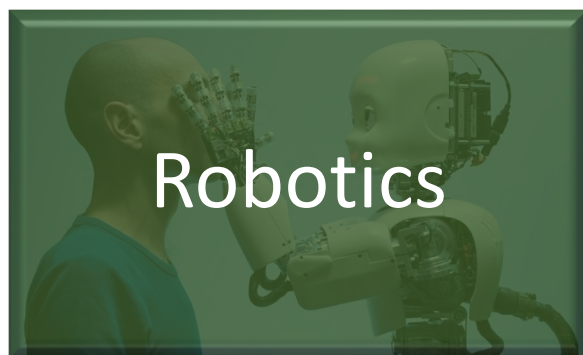
Healthcare



Trading



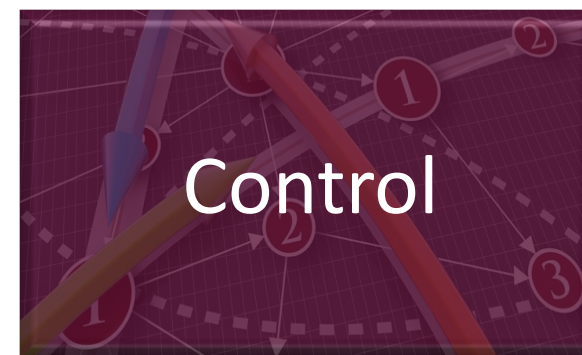
NLP



Robotics



Education



Control

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 than 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.

Trading

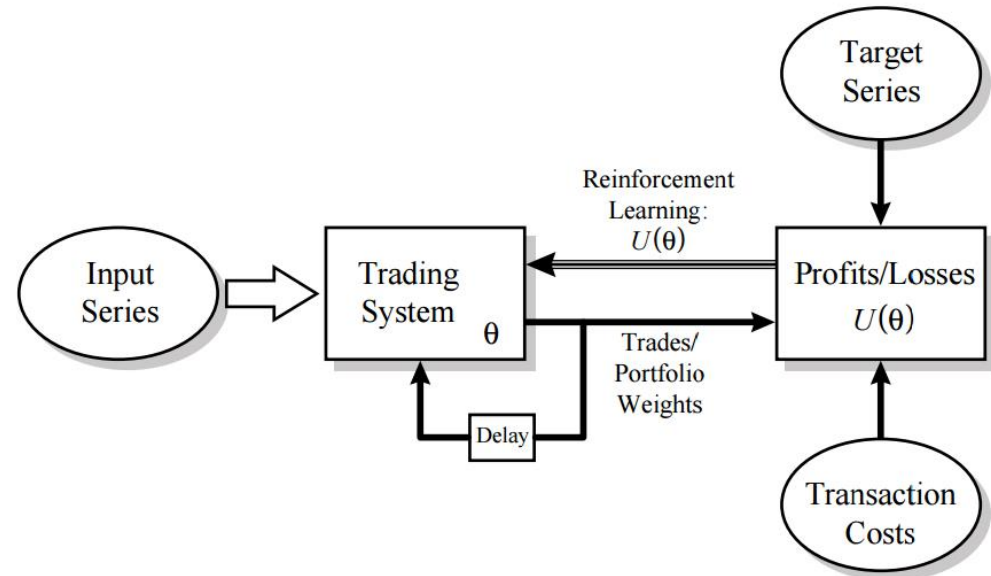
- Special Reward Target for Trading: Sharpe Ratio

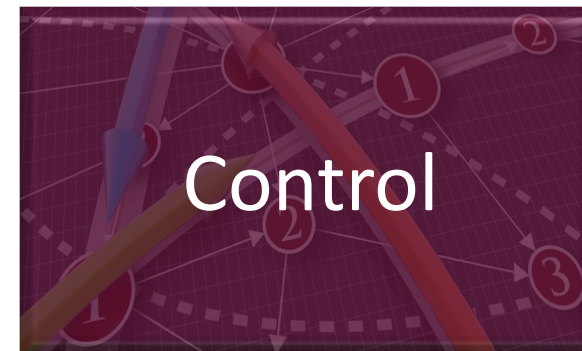
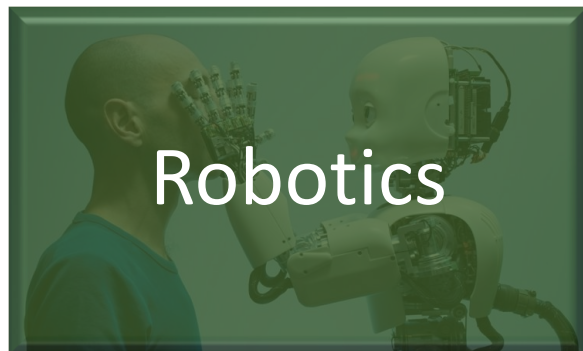
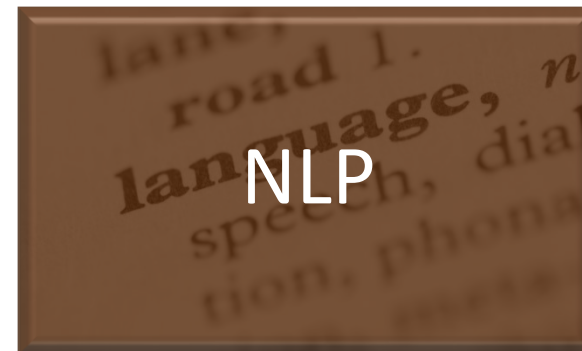
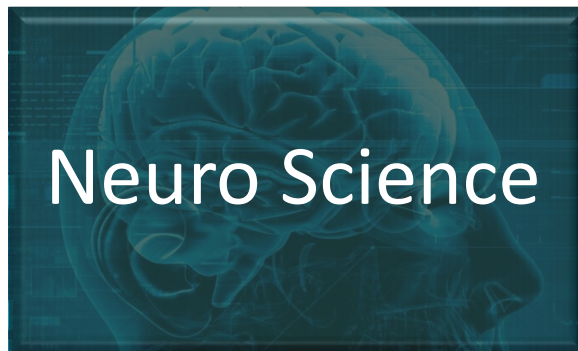
$$S_T = \frac{\text{Average}(R_t)}{\text{Standard Deviation}(R_t)}$$

- Recurrent Reinforcement Learning
 - specially tailored policy gradient

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

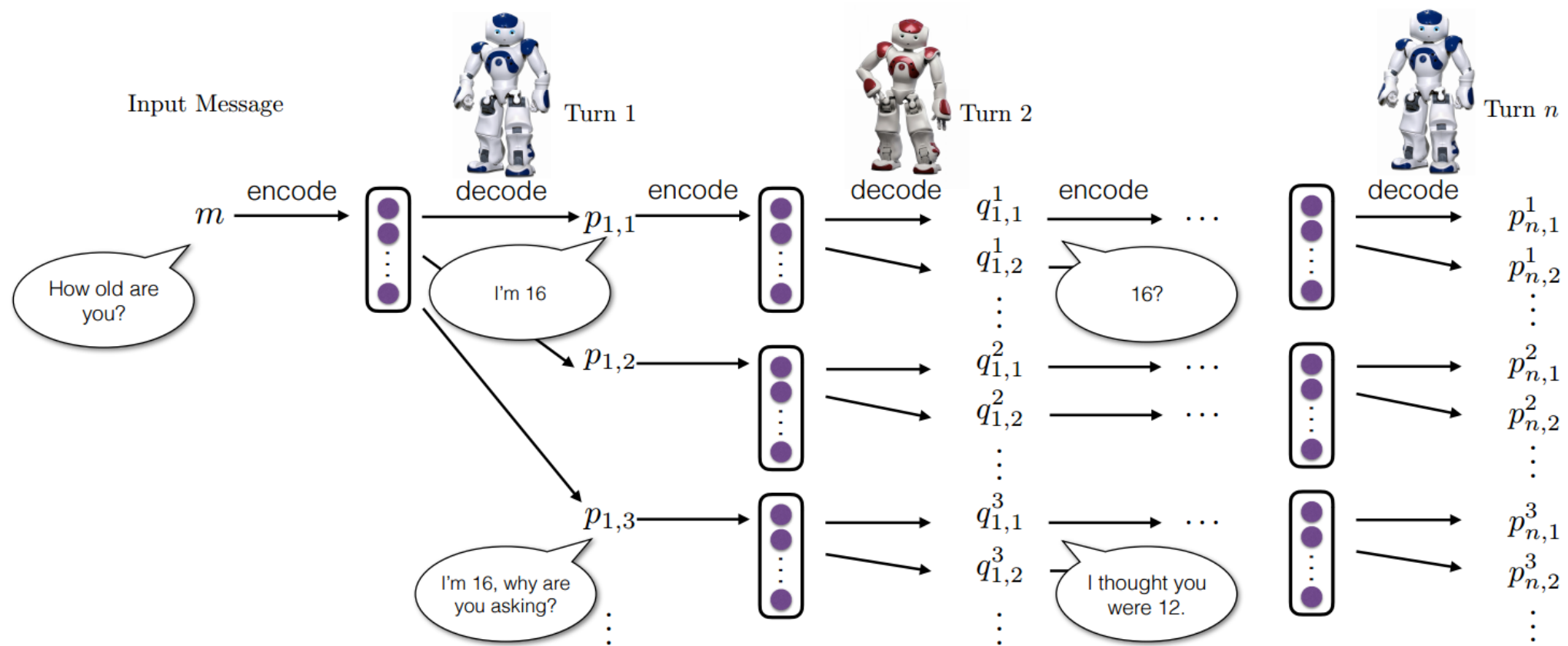
$$\Delta\theta_t = \rho \frac{dU_t(\theta_t)}{d\theta_t}$$





Natural Language Processing

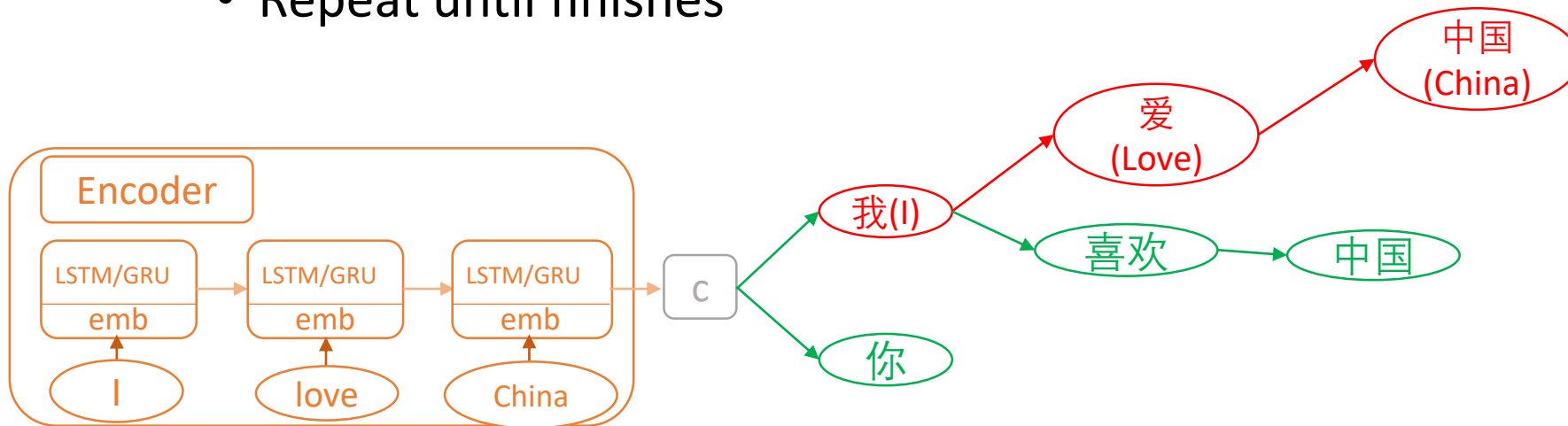
- Conversational agents



| Baseline mutual information model (Li et al. 2015) | Proposed reinforcement learning model |
|--|--|
| <p>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) </p> | <p>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) ...</p> |
| <p>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) ...</p> | <p>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) </p> |

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



Value Network- training and inference

- For each bilingual data pair (x,y) , and a translation model from $X \rightarrow Y$
 - 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

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



Game



Neuro Science



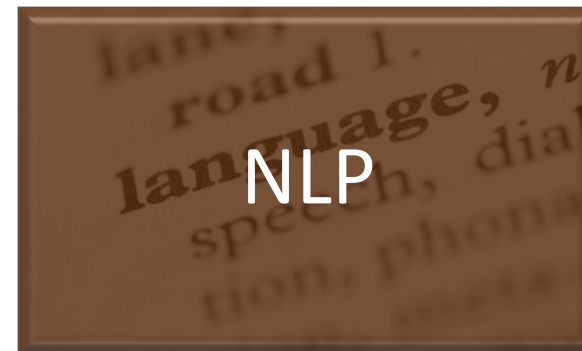
Music & Movie



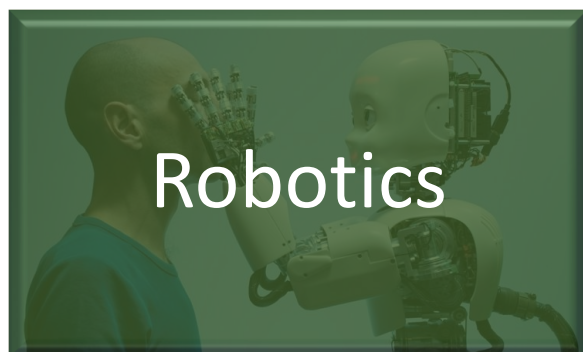
Healthcare



Trading



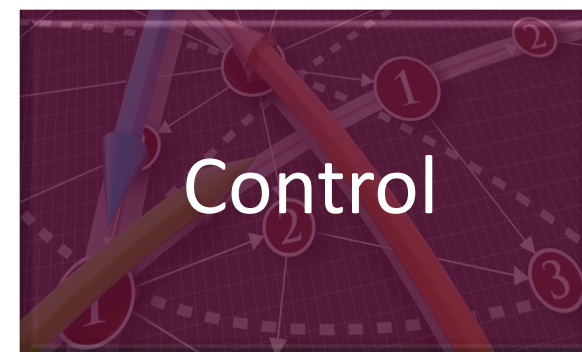
NLP



Robotics



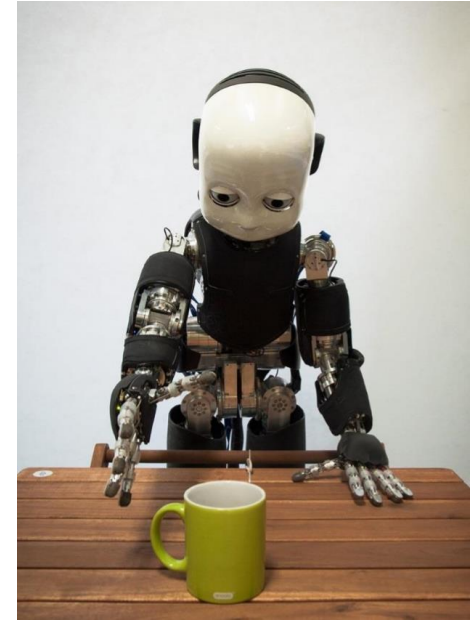
Education



Control

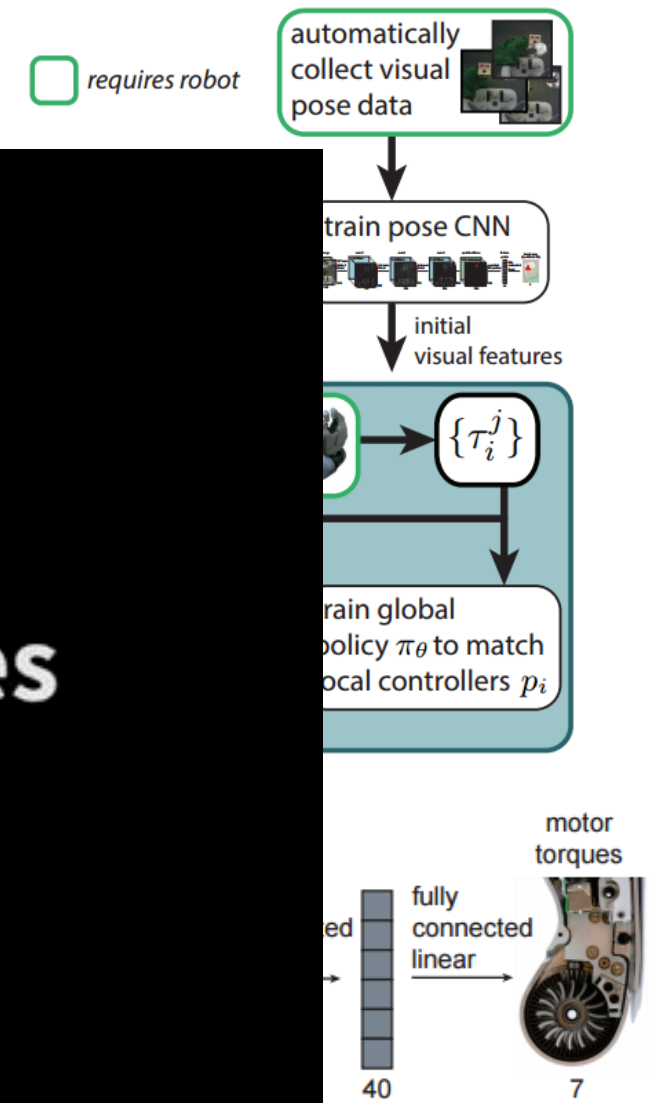
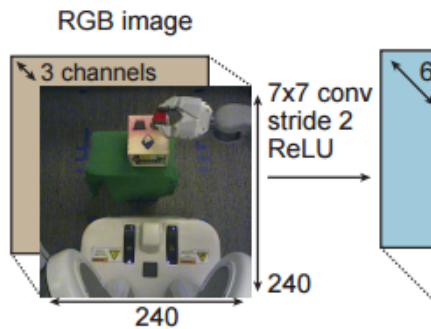
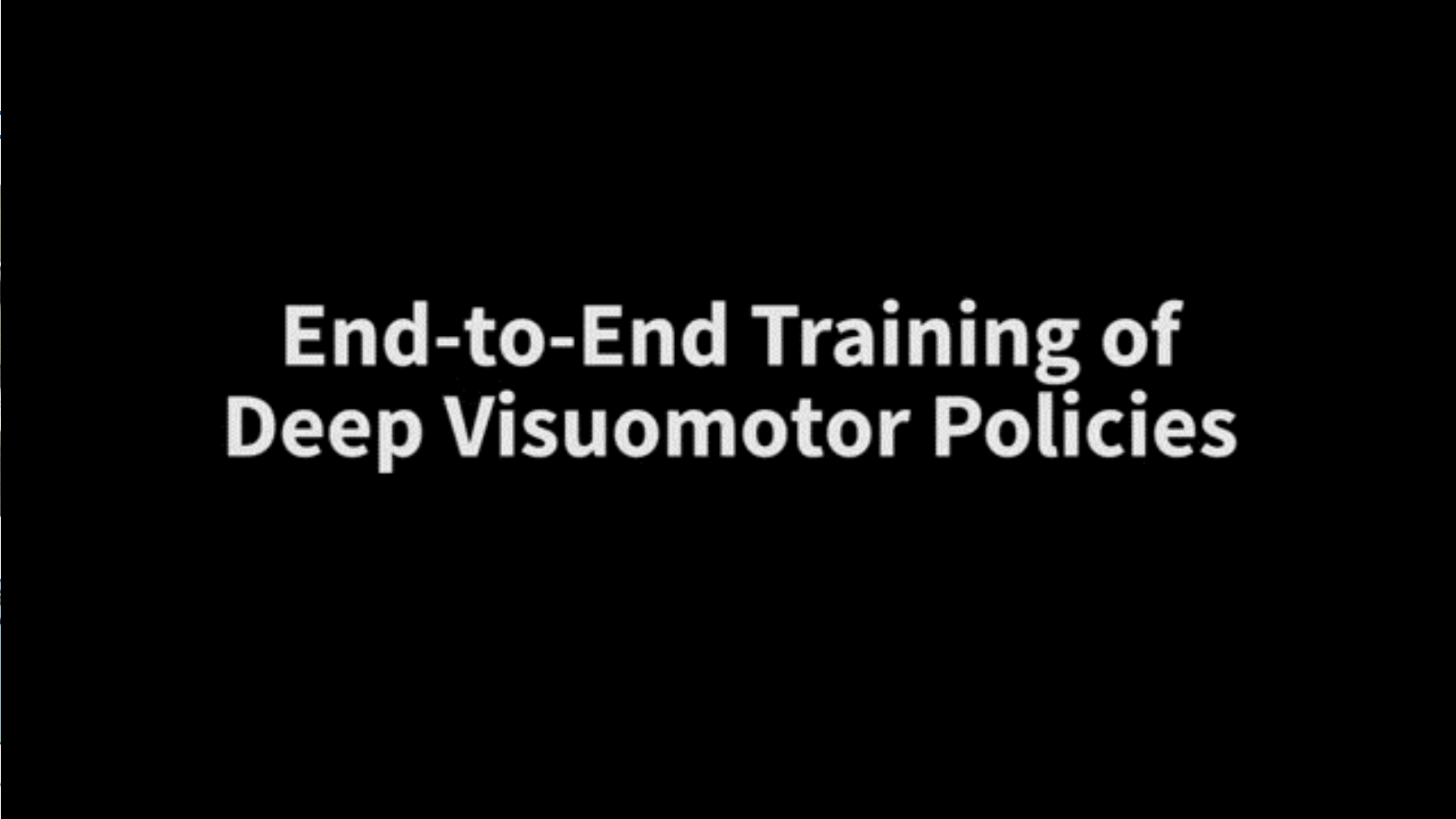
Robotics

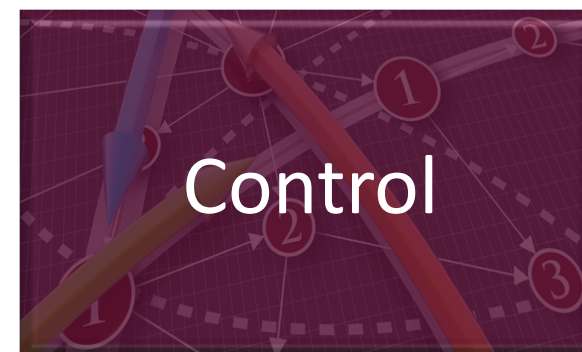
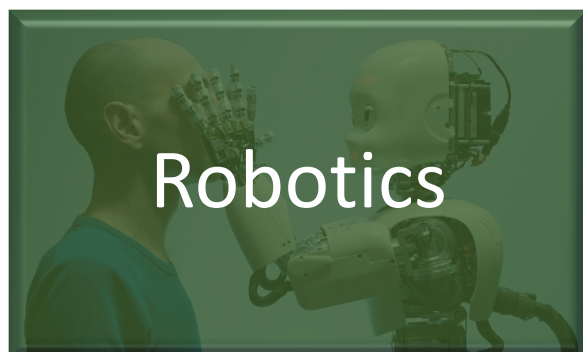
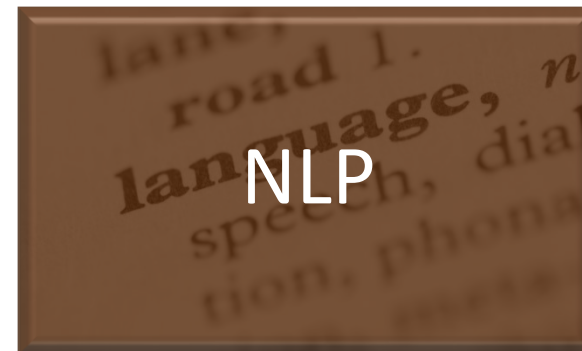
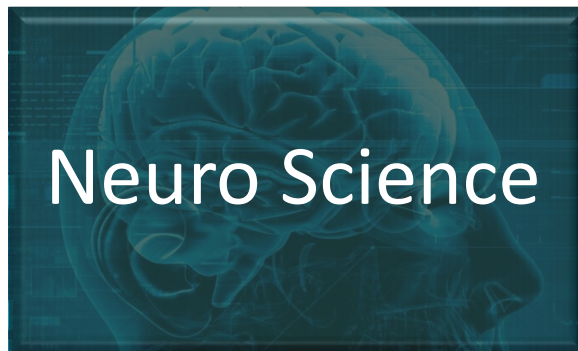
- Sequential Decision Making in Robotics



Robotics

- End-to-





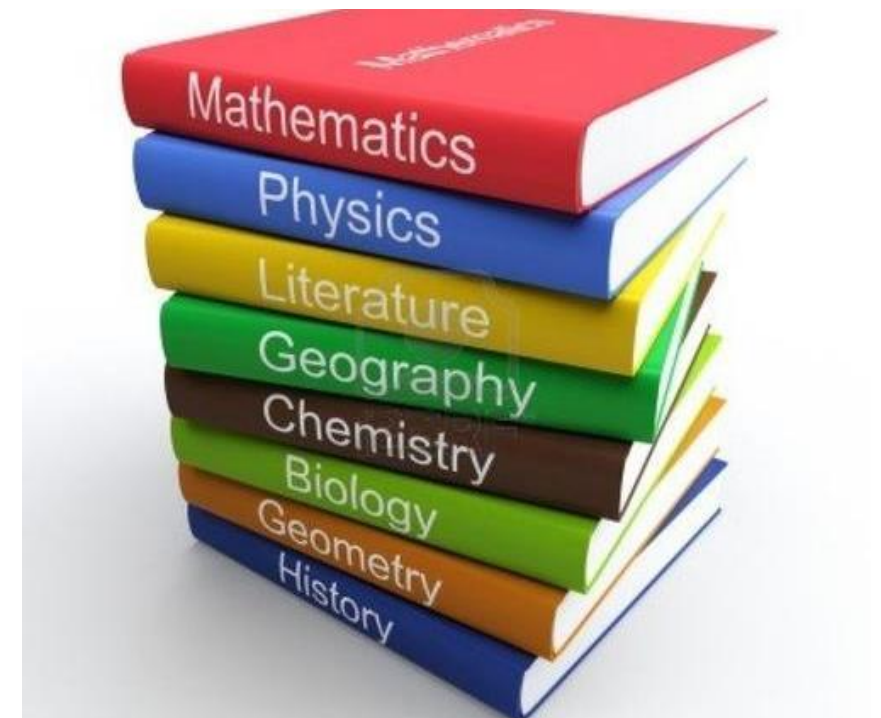
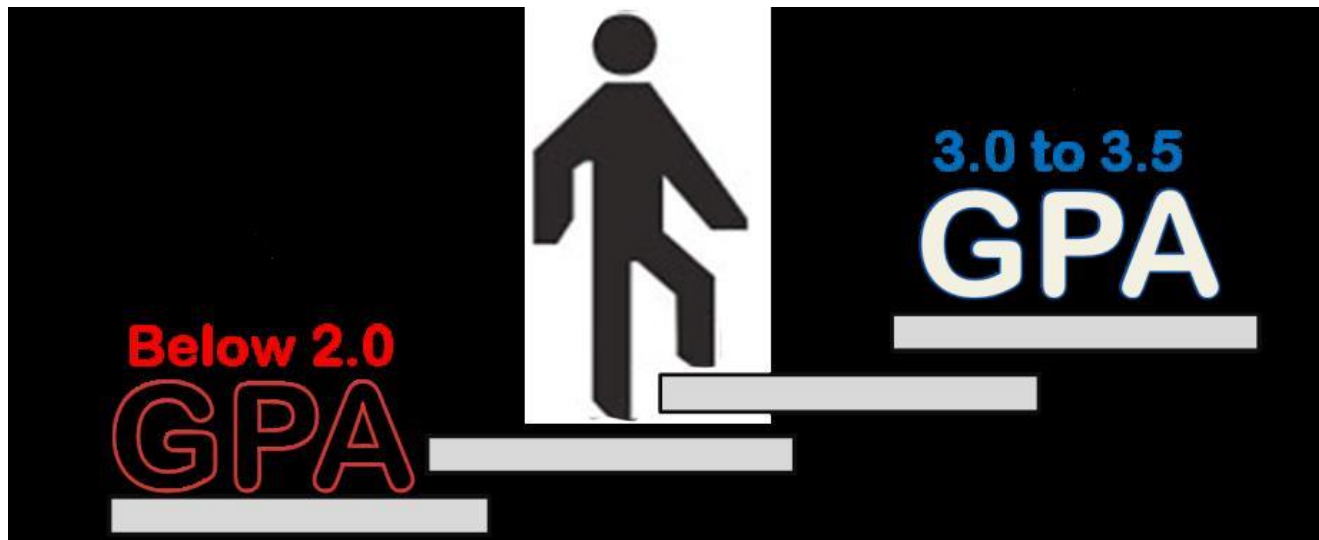
Education

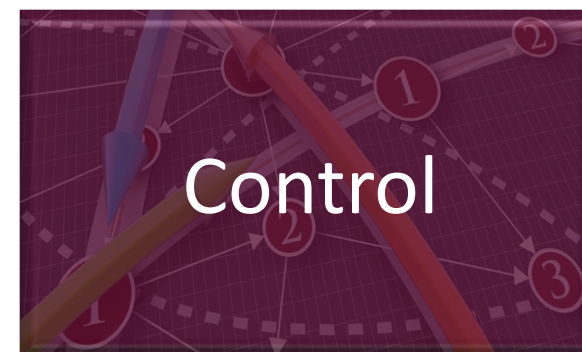
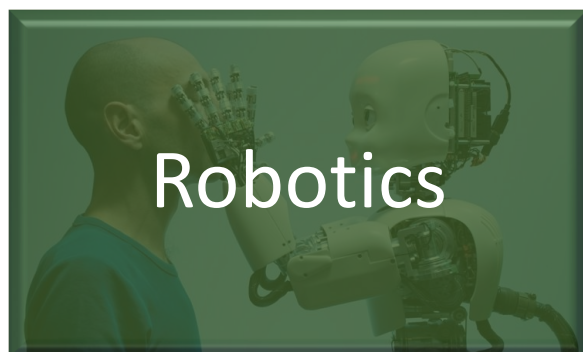
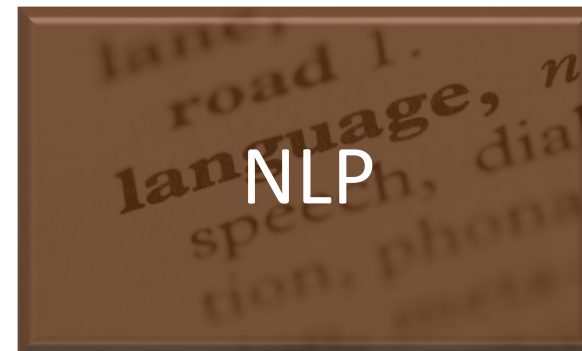
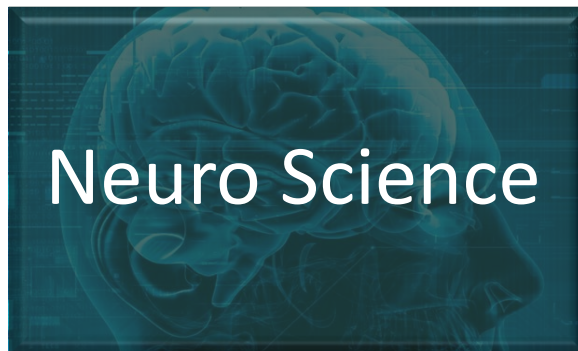
- Agents making decisions as interact with students
- Towards efficient learning



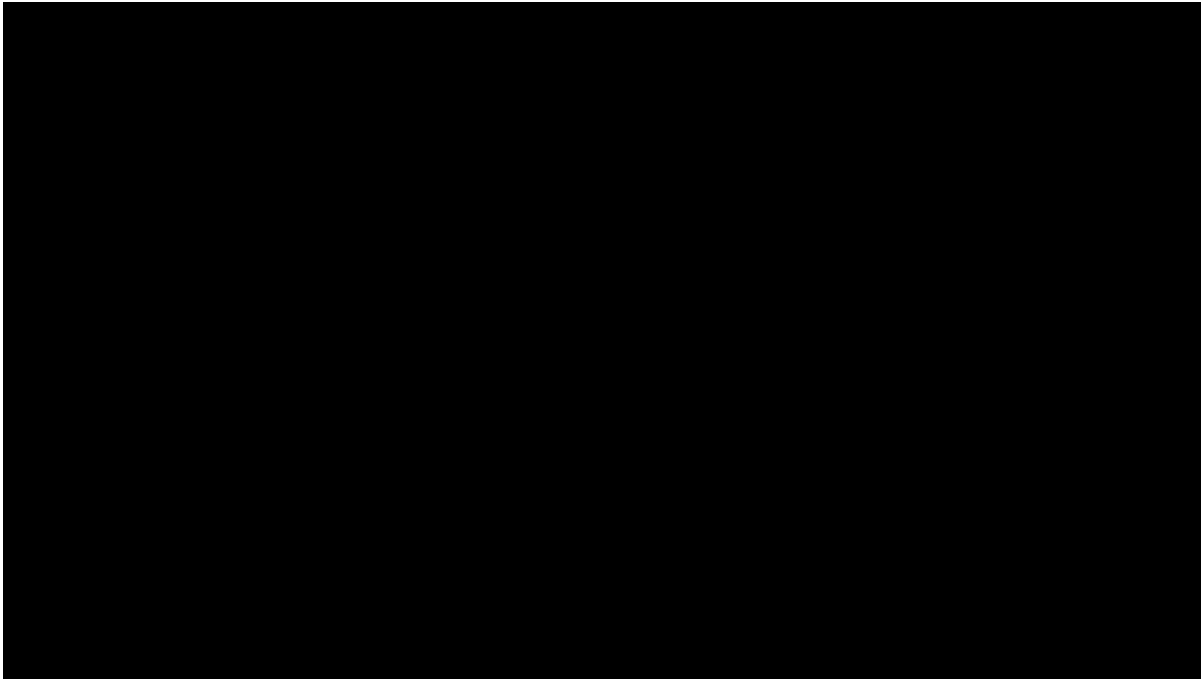
Education

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





Control



[Stanford Autonomous Helicopter](#)

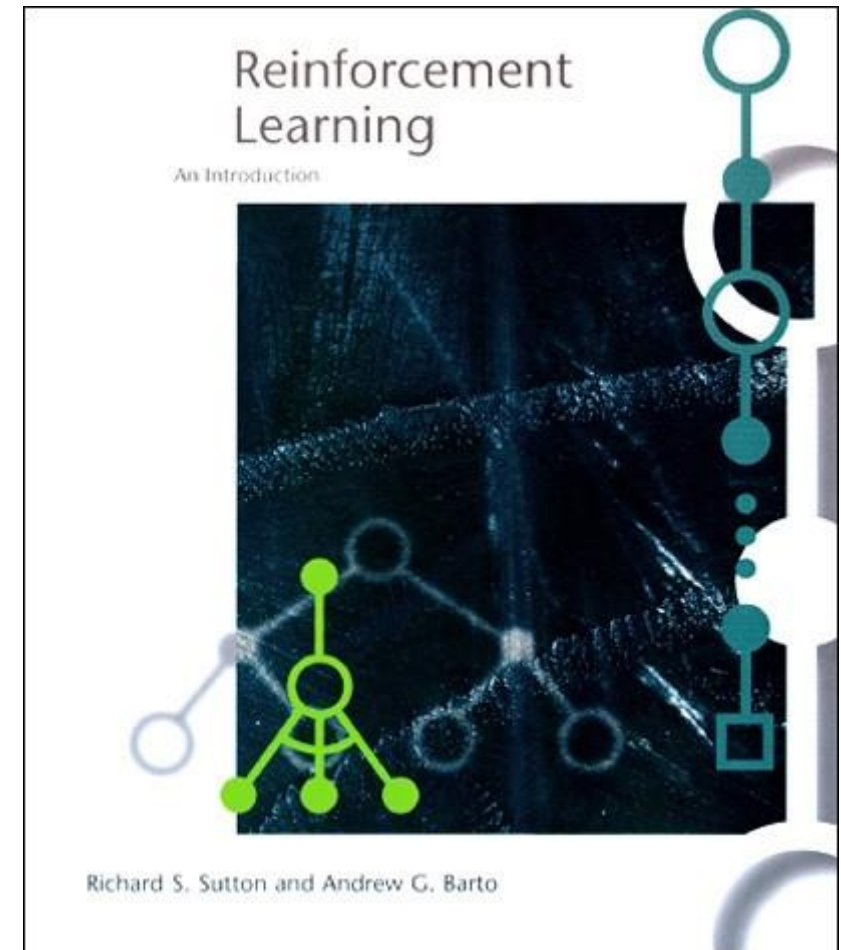


[Google's self-driving cars](#)

[Inverted autonomous helicopter flight via reinforcement learning](#), by Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger and Eric Liang. In International Symposium on Experimental Robotics, 2004.

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://rl.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>



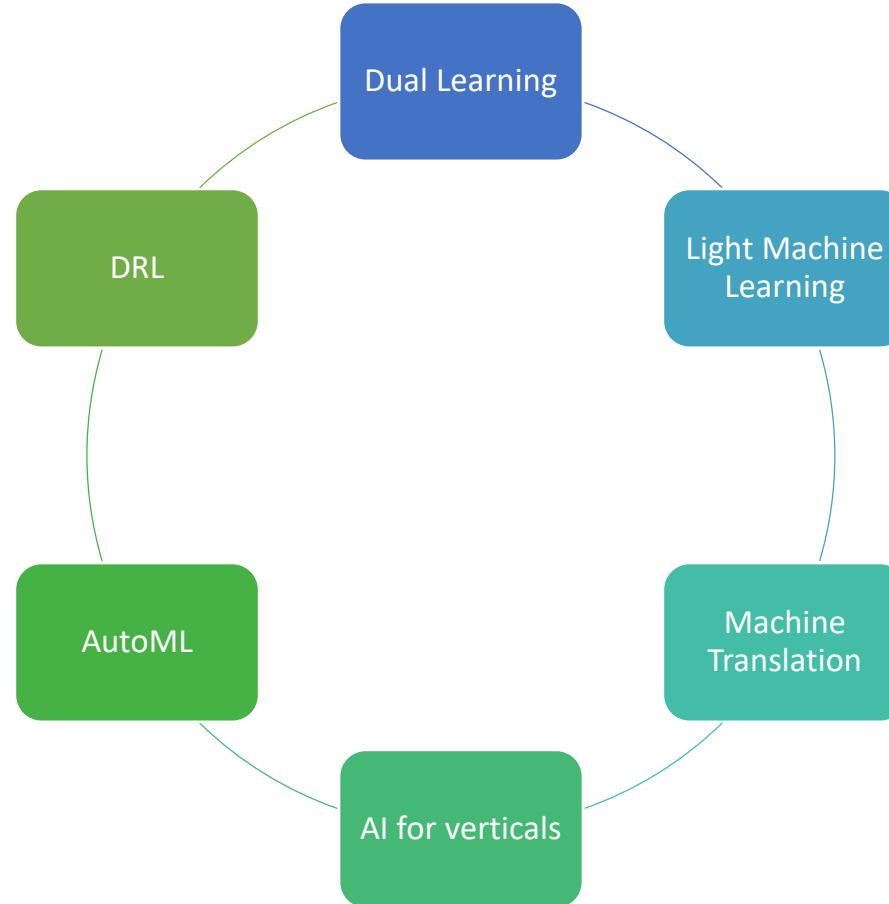
Acknowledgements

- Some content borrowed from David Silver's lecture
- My colleagues Li Zhao, Di He
- My interns Zichuan Lin, Guoqing Liu

Our Research

- Leverage symmetric structure of AI tasks to enhance learning
- Dual learning from unlabeled data, dual supervised learning, dual inference

- Robust and efficient algorithms
- Imperfect-information games



- LightRNN, LightGBM, LightLDA, LightNMT
- Reduce the model size, improve the training efficiency

- Self-tuning/learning machine
- Reinforcement learning for hyper parameter turning and training process automation

- Advanced learning/inference strategies
- New model architectures
- Low-resource translation

- Enhance all industries (e.g., finance, insurance, logistics, education...) with deep learning and reinforcement learning
- Collaboration with external partners

We are hiring!
Welcome to join us!!!



taoqin@microsoft.com



<http://research.microsoft.com/users/taoqin/>

Thanks!

taoqin@Microsoft.com