

# **Brain Inspired Artificial Intelligence**

## **1: Introduction to Reinforcement Learning**

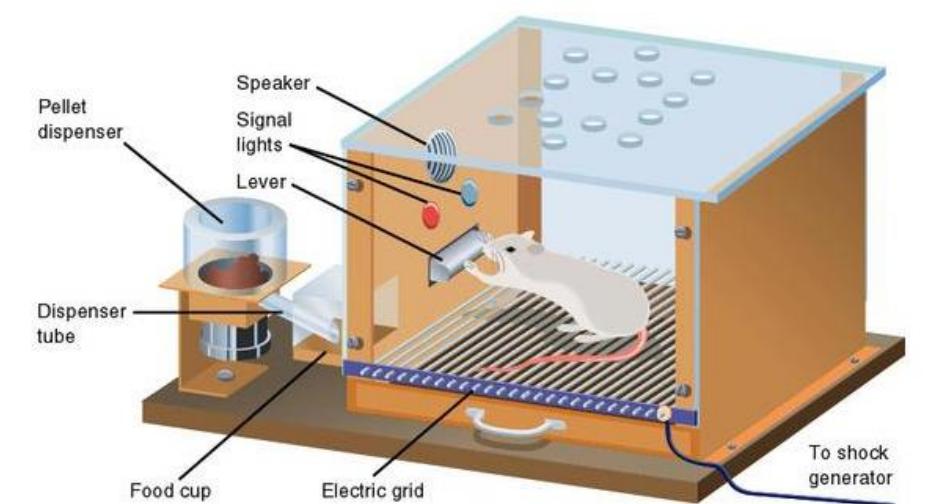
Eiji Uchibe

Dept. of Brain Robot Interface

ATR Computational Neuroscience Labs.

# What is Reinforcement Learning (RL)?

- RL is a computational framework to find an optimal policy (controller) by **trial and error**
- Inspired from psychology
  - Thorndike's law of effect
  - Skinner's principle of reinforcement
- The most famous application is **AlphaGo** that has defeated top Go masters from across the world
- Learn behaviors from **rewards** designed by each trainer



[Simply Psychology]

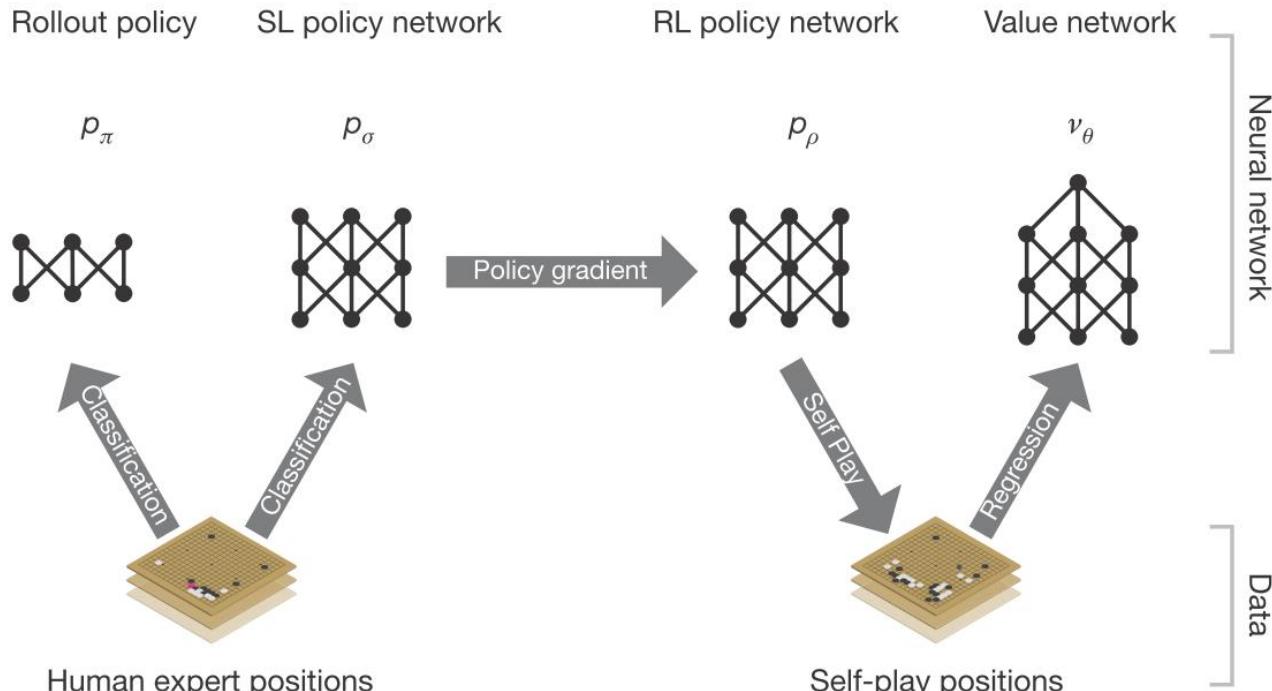


[gizmodo.jp]

# AlphaGo (Go)

- Combination of
  - imitation learning
  - policy gradient reinforcement learning
  - value based learning

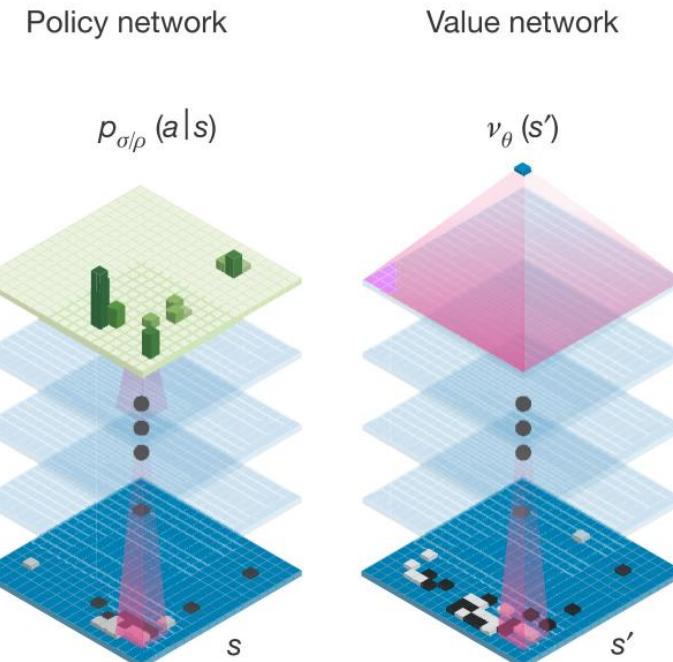
a



Go ratings

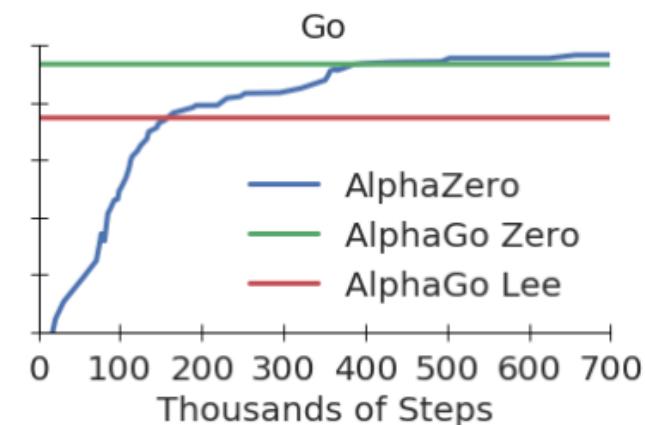
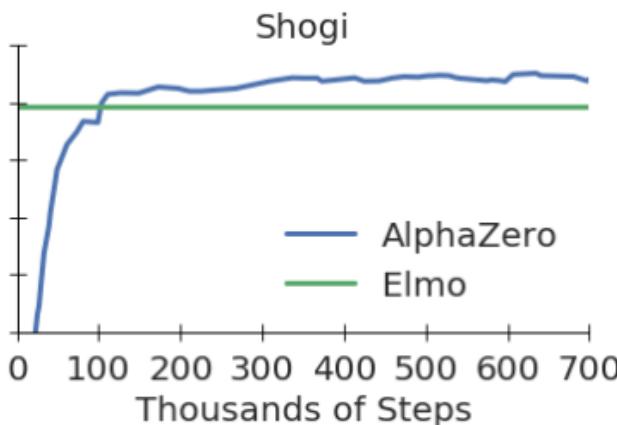
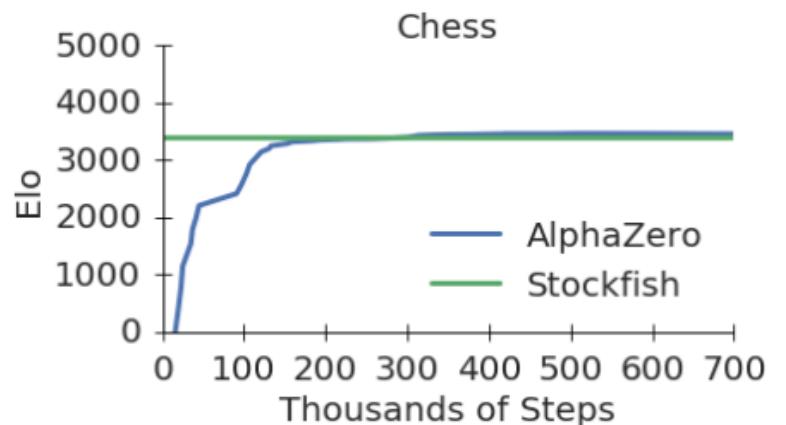
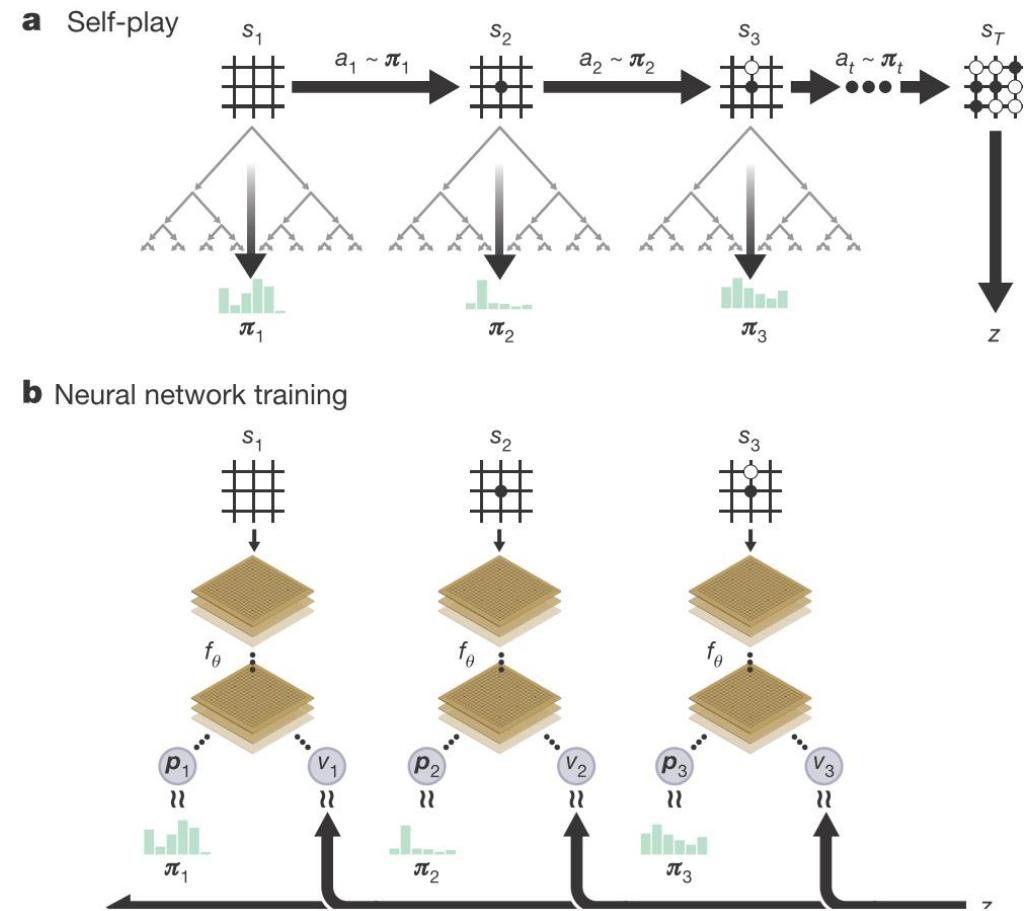
Rank	Name	♂ ♀	Flag	Elo
1	<a href="#">Ke Jie</a>	♂		3615
2	<a href="#">Google AlphaGo</a>			3585
3	<a href="#">Park Jungwhan</a>	♂		3569
4	<a href="#">Iyama Yuta</a>	♂		3532
5	<a href="#">Lee Sedol</a>	♂		3519

b



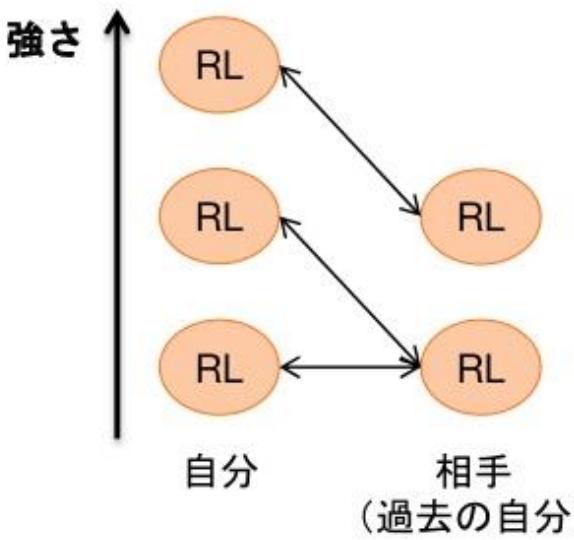
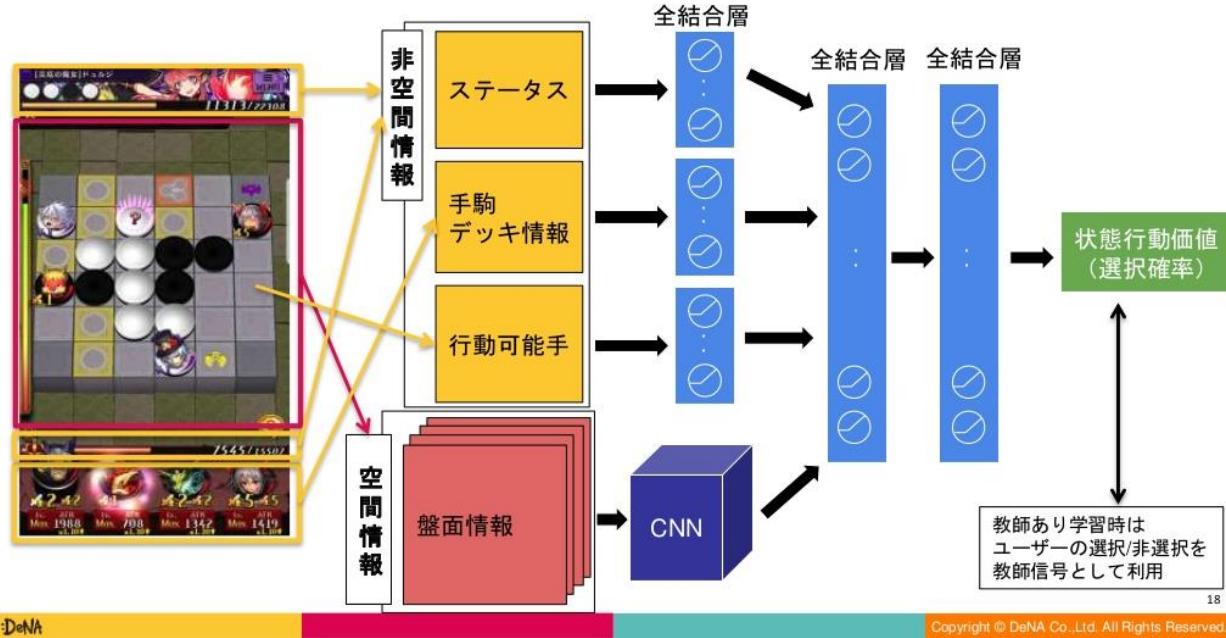
# Extension of AlphaGo

- AlphaGo Zero [Silver et al., 2017a]
  - self-play, without human play data
  - integrate policy and value network
- AlphaZero [Silver et al., 2017b]
  - chess, shogi, and go



# Playing a video game

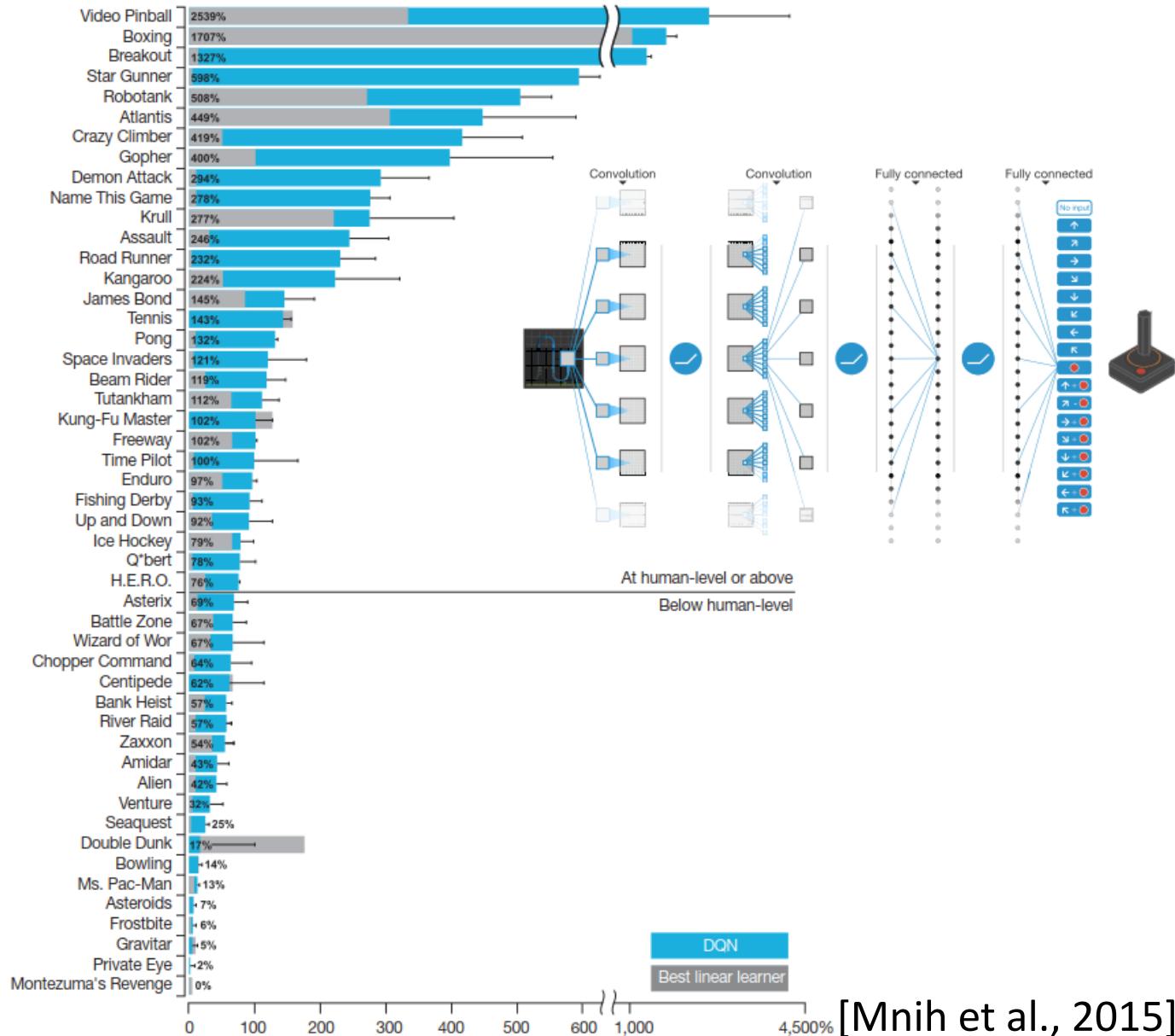
- RL is applied to a Othello-like game (逆転オセロニア)
- Supervised learning
  - NN architecture is similar to that of DQN
  - use top players data
- Reinforcement learning
  - self-play



<https://youtu.be/p-iCmOo3gUo>  
<https://fullswing.dena.com/othellonia-ai/>  
<https://www.slideshare.net/juneokumura/ss-82644332>

# Deep Q Net (Play Video Games)

- Rules of the game are unknown
- Integrate RL with deep learning
- Learn directly from a sequence of game screens



# Helicopter Control

- Leverage expert demonstrations to efficiently learn good controllers for tasks being demonstrated by an expert

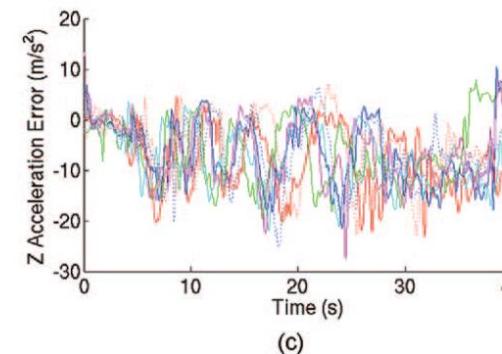
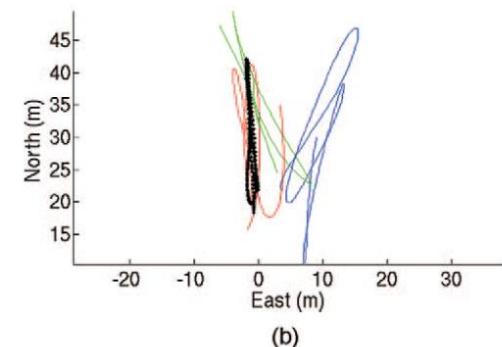
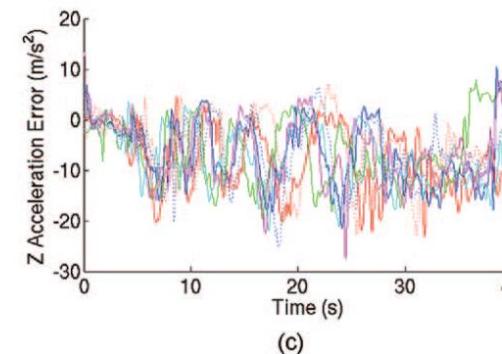
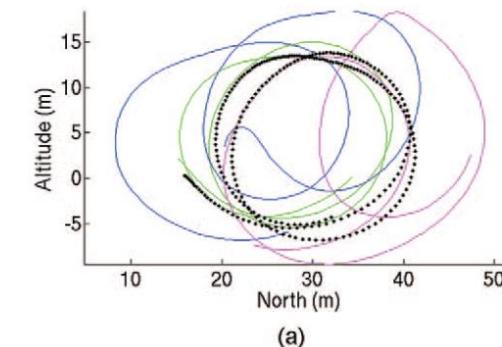
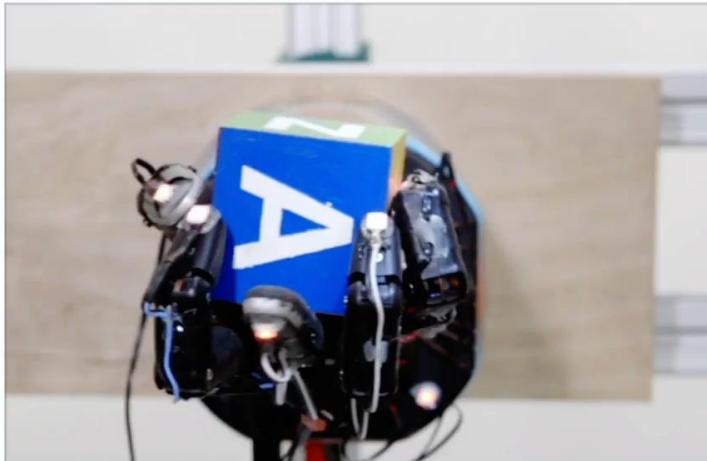
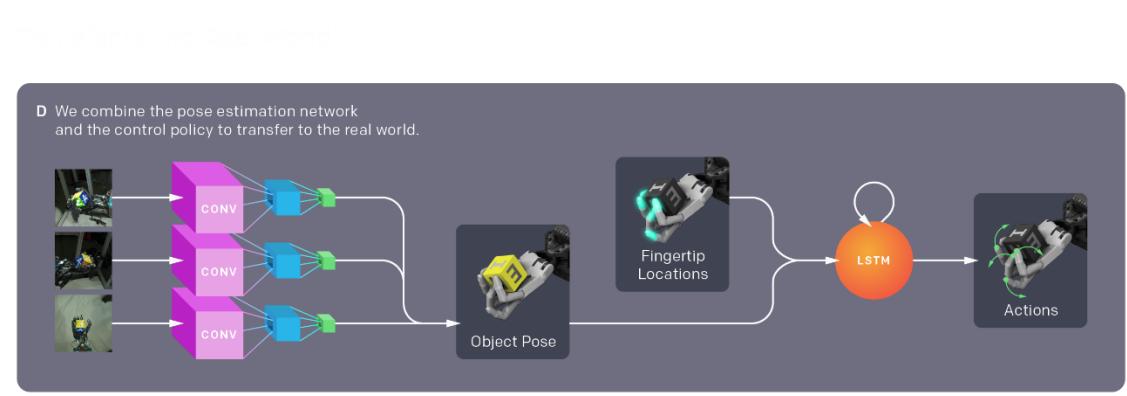
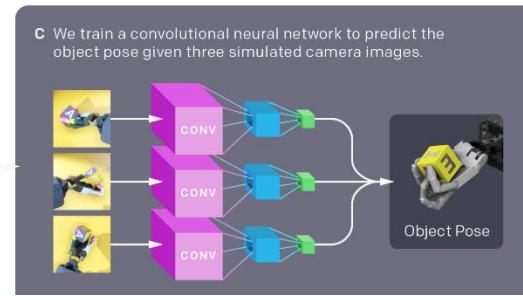
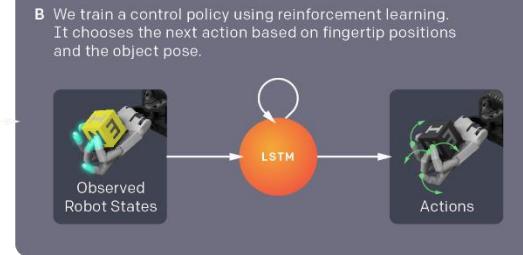


Fig. 6. Colored lines: demonstrations. Black dotted line: trajectory inferred by our algorithm. (See the text for details.)

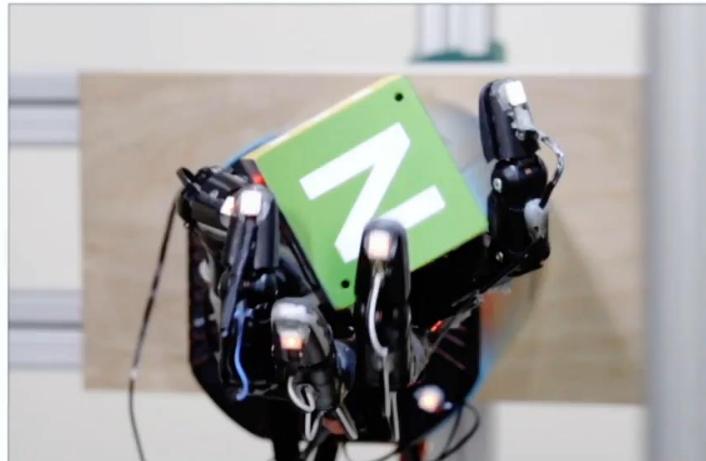
[Abbeel et al., 2010]

# In-hand manipulation

[Andrychowicz et al., 2018]



FINGER PIVOTING



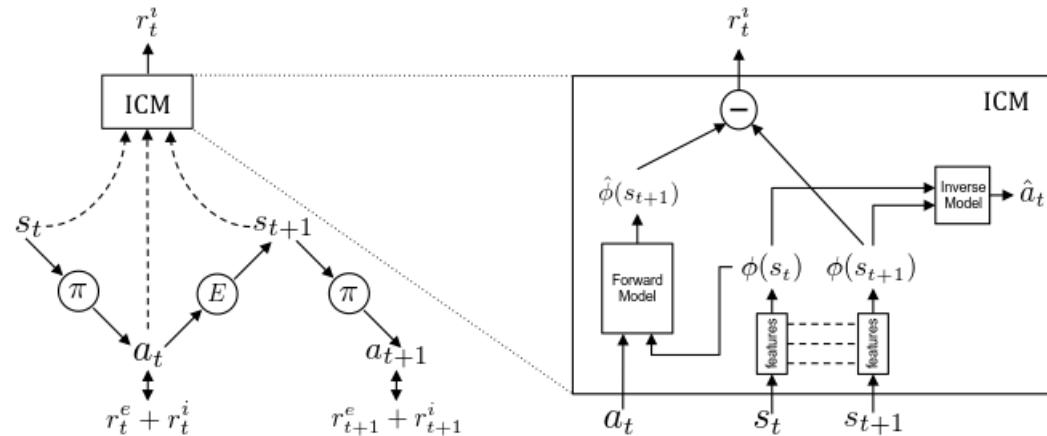
SLIDING



FINGER GAITING

# Curiosity-Driven Reinforcement Learning

Curiosity Driven Exploration  
by Self-Supervised Prediction



ICML 2017

Deepak Pathak, Pulkit Agrawal, Alexei Efros, Trevor Darrell  
UC Berkeley

Level Ids	Level-1		Level-2				Level-3			
	Scratch 1.5M	Run as is 0	Fine-tuned 1.5M	Scratch 1.5M	Scratch 3.5M	Run as is 0	Fine-tuned 1.5M	Scratch 1.5M	Scratch 5.0M	
Mean $\pm$ stderr	$711 \pm 59.3$	$31.9 \pm 4.2$	$466 \pm 37.9$	$399.7 \pm 22.5$	$455.5 \pm 33.4$	$319.3 \pm 9.7$	$97.5 \pm 17.4$	$11.8 \pm 3.3$	$42.2 \pm 6.4$	
% distance > 200	$50.0 \pm 0.0$	0	$64.2 \pm 5.6$	$88.2 \pm 3.3$	$69.6 \pm 5.7$	$50.0 \pm 0.0$	$1.5 \pm 1.4$	0	0	
% distance > 400	$35.0 \pm 4.1$	0	$63.6 \pm 6.6$	$33.2 \pm 7.1$	$51.9 \pm 5.7$	$8.4 \pm 2.8$	0	0	0	
% distance > 600	$35.8 \pm 4.5$	0	$42.6 \pm 6.1$	$14.9 \pm 4.4$	$28.1 \pm 5.4$	0	0	0	0	

# Tax Collections Optimizers

- Optimally managing the tax collection processes at financial institutions
- Use actually in New York State Department of Taxation and Finance

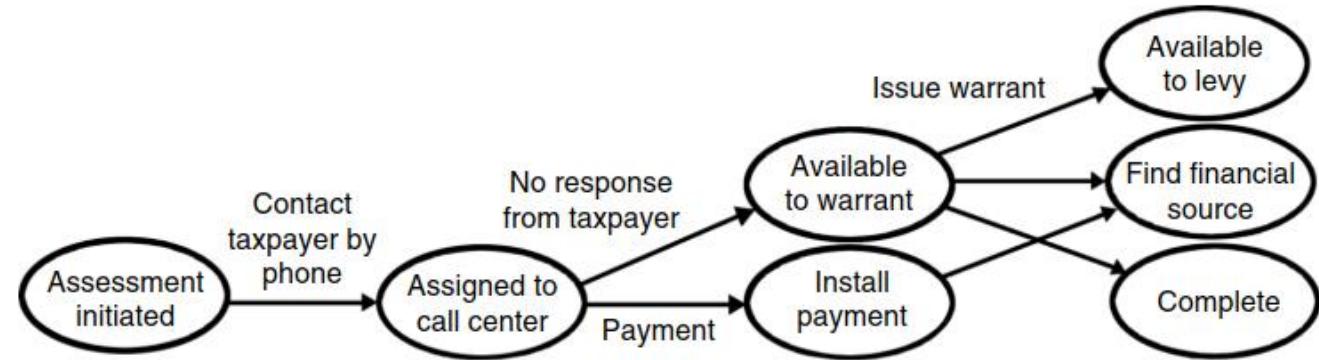
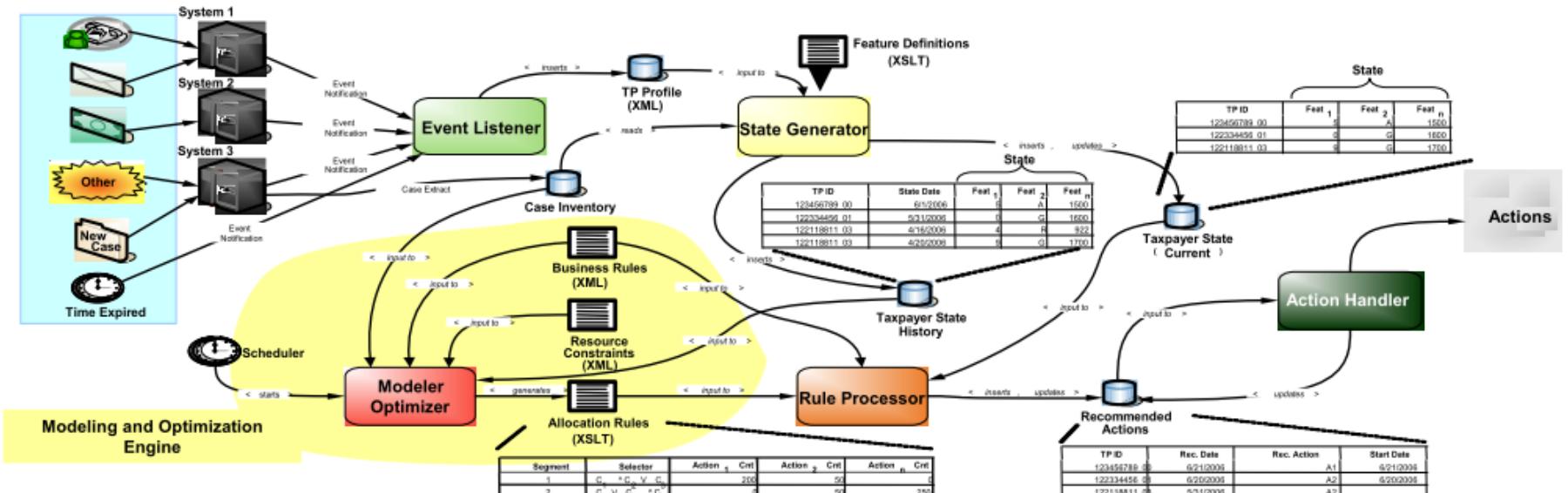


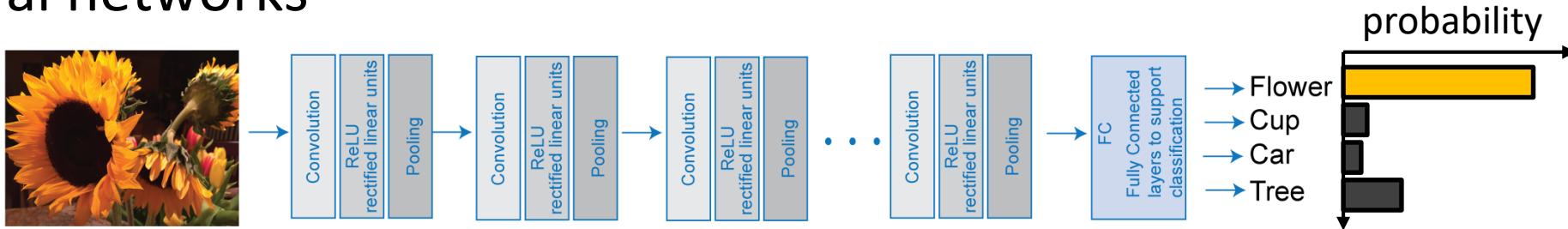
Figure 1: The graphic illustrates an MDP formulation of the collections process.



# Difference between Deep Learning and (Deep) Reinforcement Learning

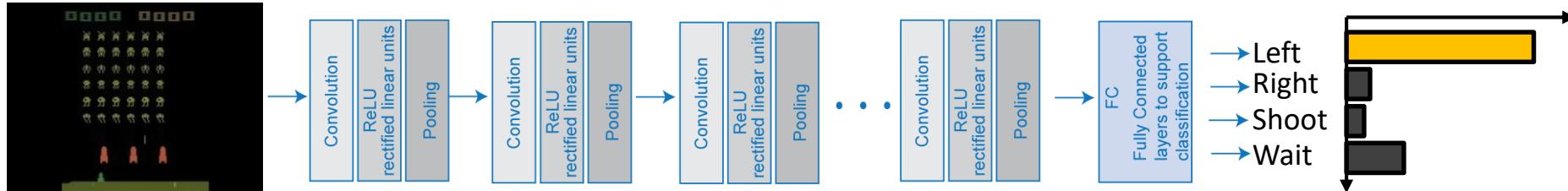
- Deep Learning for Classification

Take input data and predict a category for that data by using multilayer neural networks



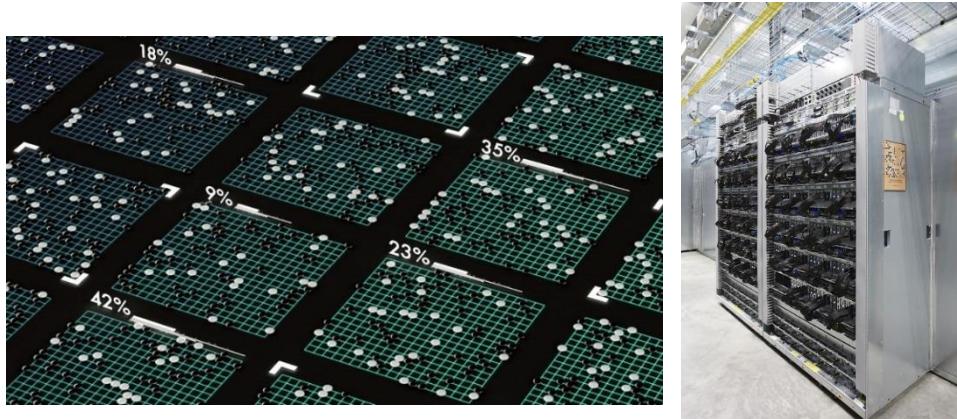
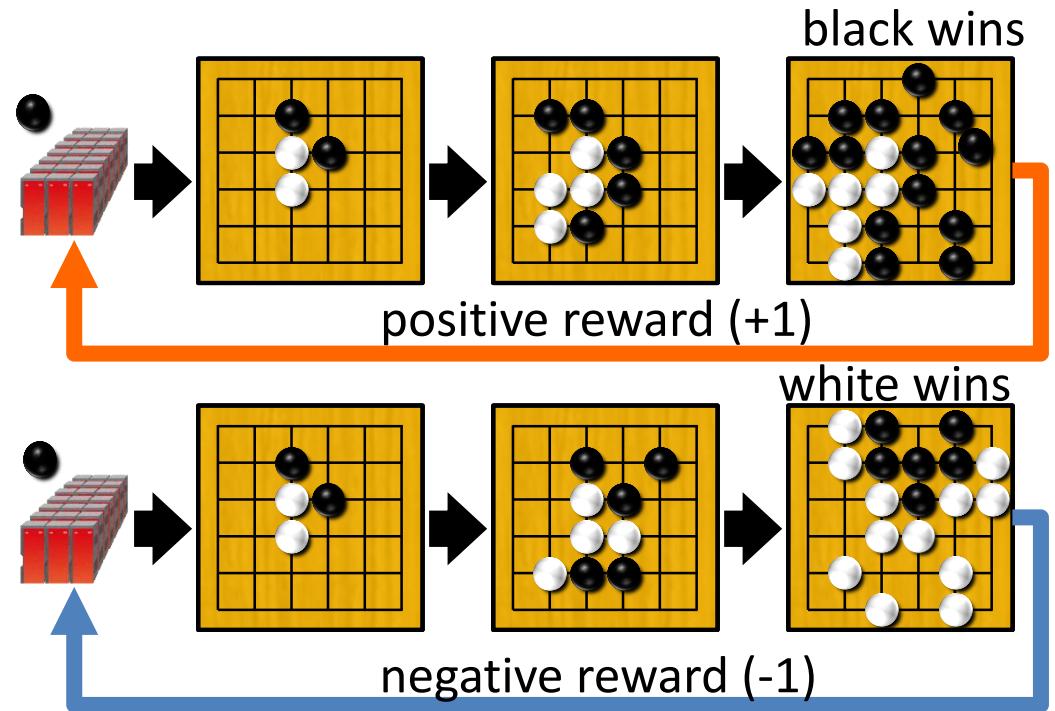
- Deep Reinforcement Learning

Find a policy that maximizes the sum of rewards by trial and error

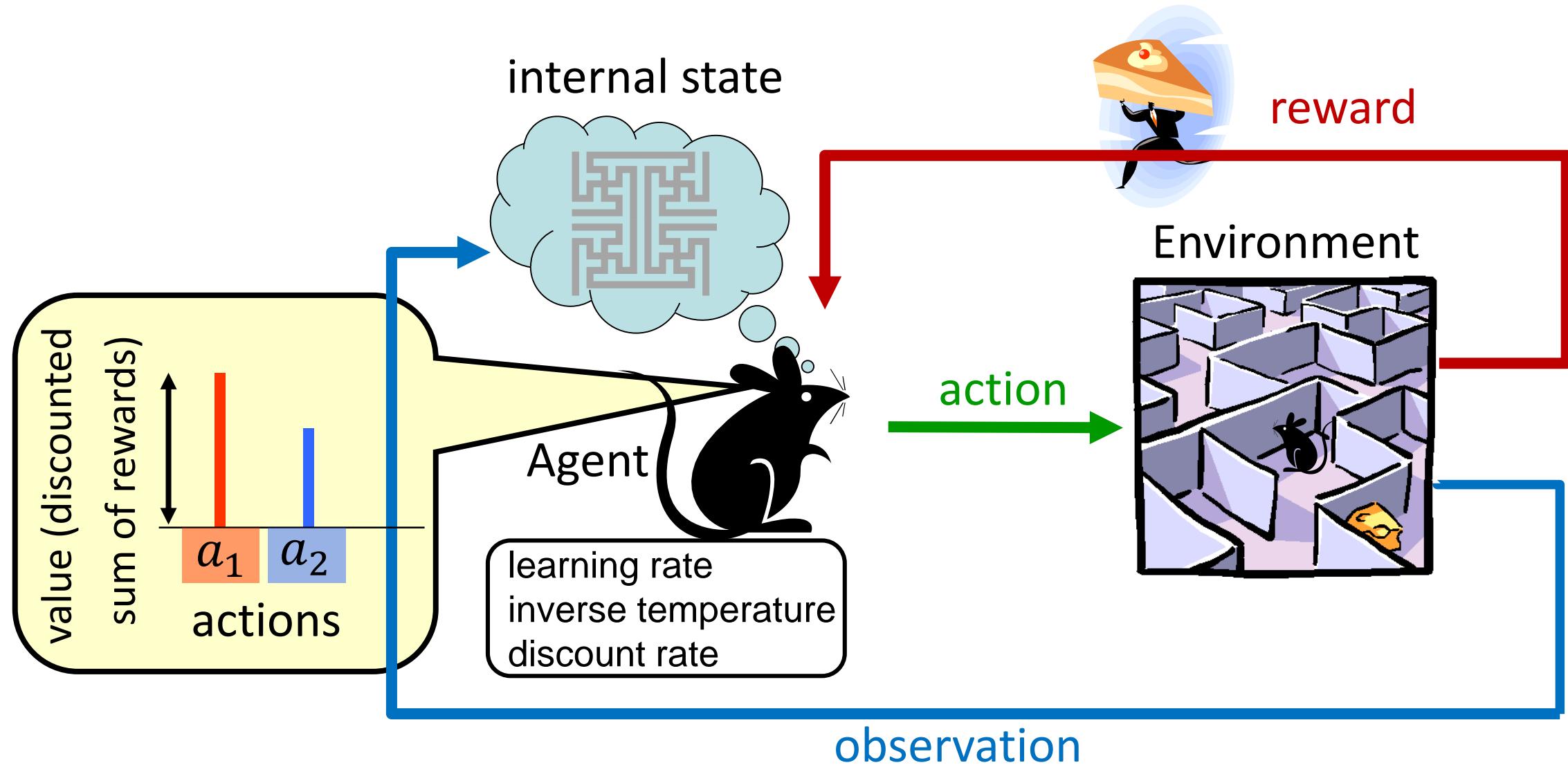


# What is Reward?

- In the case of Go
  - positive reward for winning
  - negative reward for losing
  - zero otherwise
- very sparse reward
- AlphaGo Zero, which does not use a record of a game of go, needs **4.9 million** games of self-play
- Deep RL is applicable when we can collect samples by using multiple simulators

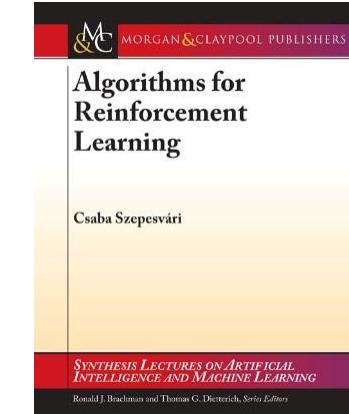


# Interaction between Agent and Environment



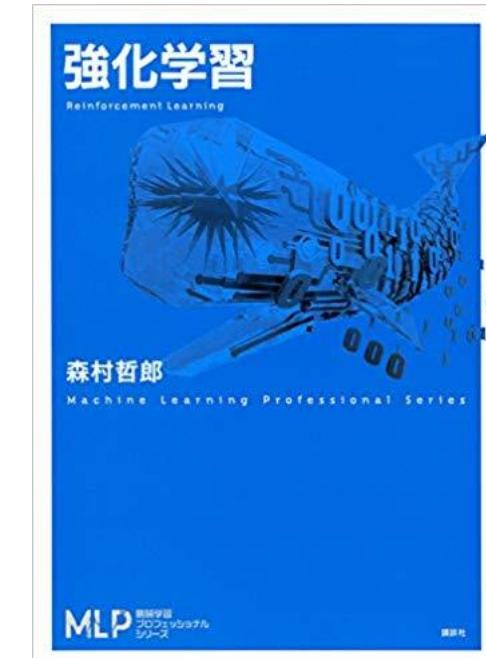
# Textbooks

- Sutton, R.S. and Barto, A.G.  
Reinforcement learning: An introduction.  
MIT Press
  - [First edition](#) (1998), Japanese translation (2000), and [Second edition](#) (2018).
  - Must-read for researchers
- Szepesvári, C. (2010). [Algorithms for reinforcement learning](#). Morgan & Claypool.
  - Japanese translation (2017)
  - More compact than Sutton and Barto (1998)



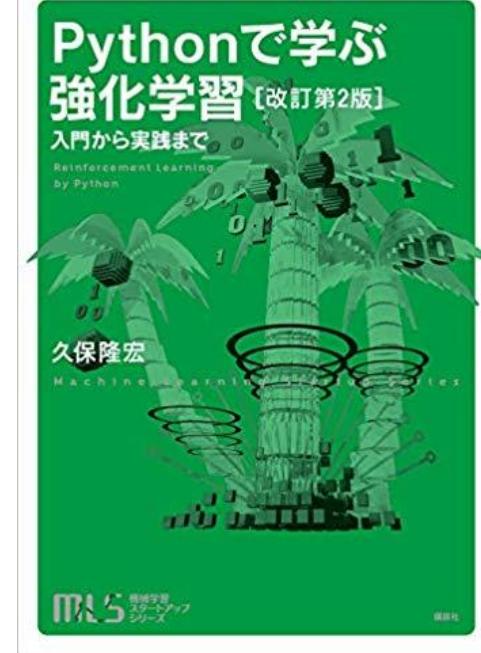
# Textbooks

- 牧野貴樹、澁谷長史(編) (2016)  
これからの強化学習. 森北出版
  - Overview of modern reinforcement learning such as policy search methods, deep RL, and inverse RL
- 森村哲郎 (2019). 機械学習  
プロフェッショナルシリーズ 強化学習
  - Theoretical introduction to reinforcement learning with discrete states and actions



# Textbooks

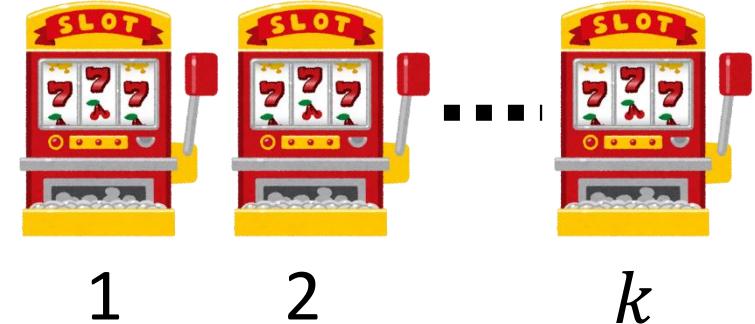
- 久保隆宏 (2019). Pythonで学ぶ強化学習 [改訂第2版]  
入門から実践まで. 講談社
  - For practitioners (samples codes)
  - Including inverse reinforcement learning
- 片平 健太郎 (2018). 行動データの計算論  
モデリング: 強化学習モデルを例として  
オーム社
  - Computational modeling of behavioral data based on reinforcement learning



# **Reinforcement Learning for Bandit Problems**

# Multi-Armed Bandit Problem

- There exist  $k$  gambling machines
- Each machine  $a \in [1, k]$  has a different unknown probability distribution  $p(r | a)$  for rewards
  - Action: select a gambling machine
- The goal is to maximize rewards obtained by successively playing gamble machines (the ‘arms’ of the bandits)



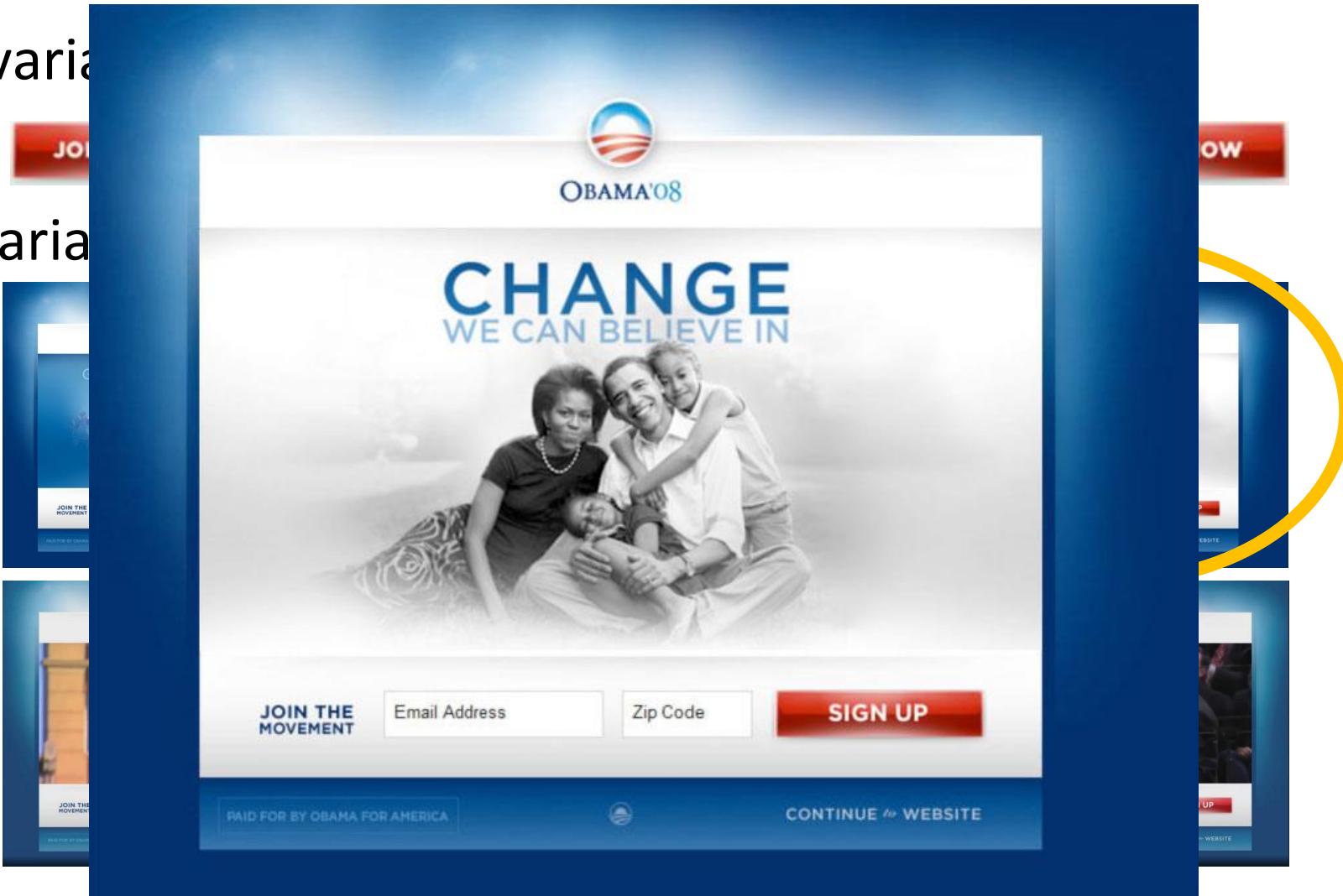
# Example: Web Site Optimization

- Former President Obama raised \$60 million dollar for the United States presidential election of 2008
- His team experimented with 2 pairs of the sign up form
  - 4 button variations
  - 6 media variations
- They tested  $4 \times 6 = 24$  forms



# Example: Web site optimization

- 4 button variations



<https://www.mailmunch.co/blog/ab-testing-got-obama-60-million/>

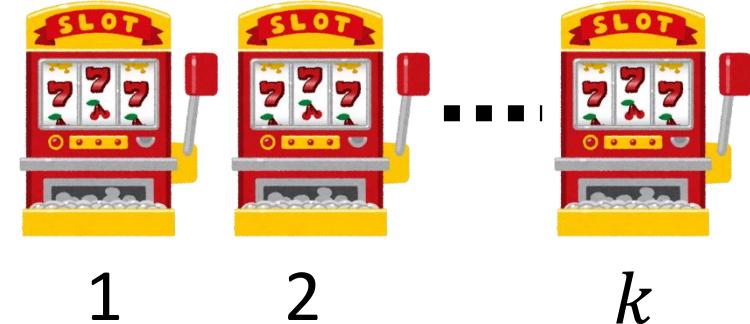
# Example: Web site optimization

- The winning variation had a sign-up rate of 11.6% as compared to the original page which had 8.26%
- This test is a special case of bandit problems

Combinations (24)		Page Sections (2)		Download: XML CSV TSV Print		
Relevance Rating	Variation	Est. conv. rate		Chance to Beat Orig.	Observed Improvement	Conv./Visitors
<b>Button</b>  5 / 5	Original	7.51% ± 0.2%		—	—	5851 / 77858
	Learn More	8.91% ± 0.2%		100%	18.6%	6927 / 77729
	Join Us Now	7.62% ± 0.2%		73.5%	1.37%	5915 / 77644
	Sign Up Now	7.34% ± 0.2%		13.7%	-2.38%	5660 / 77151
<b>Media</b>  5 / 5	Original	8.54% ± 0.2%		—	—	4425 / 51794
	Family Image	9.66% ± 0.2%		100%	13.1%	4996 / 51696
	Change Image	8.87% ± 0.2%		92.2%	3.85%	4595 / 51790
	Barack's Video	7.76% ± 0.2%		0.04%	-9.14%	3992 / 51427
	Sam's Video	6.29% ± 0.2%		0.00%	-26.4%	3261 / 51864
	Springfield Video	5.95% ± 0.2%		0.00%	-30.3%	3084 / 51811

# Components: Action and Policy

- Action
  - Choosing one gambling machine in the bandit problems
  - Discrete action
  - If there exist  $k$  machines, it is convenient to define a set of actions by  $\mathcal{A} = \{1, 2, \dots, k\}$
  - We usually assume discrete actions for simplicity

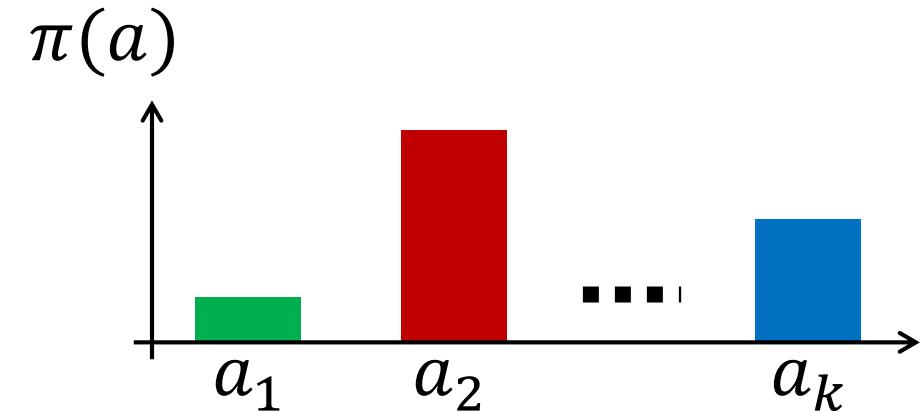


- Policy (learned by the agent)

- $\pi(a)$ : probability to select an action  $a$

- $\pi(a) \geq 0, \forall a \in \mathcal{A}$

- $\sum_{a \in \mathcal{A}} \pi(a) = 1$  →  $k - 1$  free parameters



# Components: Reward

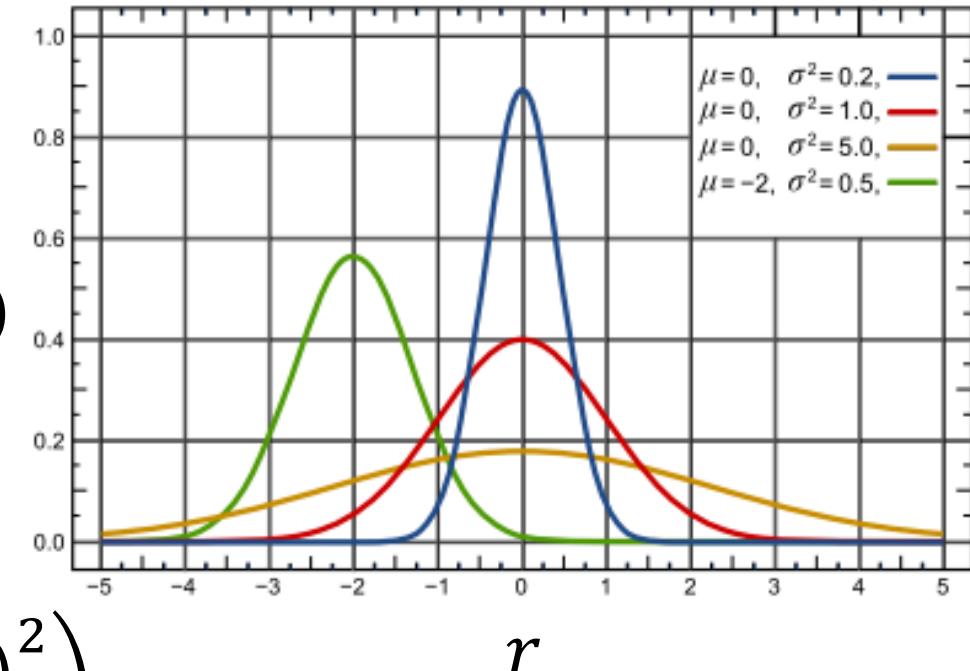
- Reward (usually unknown to a learning agent)

- When the agent executes an action  $A_t$  at time  $t$ , it receives a scalar feedback called reward

$$R_t \sim p(r | A_t)$$

- For example,  $p(r | a)$  is modeled by a Gaussian distribution with mean  $\mu(a)$  and standard deviation  $\sigma(a)$

$$p(r | a) = \frac{1}{\sqrt{2\pi}\sigma(a)} \exp\left(-\frac{(r - \mu(a))^2}{2\sigma(a)^2}\right)$$



# Goal of the Learning Agent

- Find a policy  $\pi$  that maximizes the expected reward

$$J(\pi) \triangleq \mathbb{E}_{\pi}[R] = \int rp(r \mid \pi)dr$$

- $p(r \mid \pi)$  is a probability density function

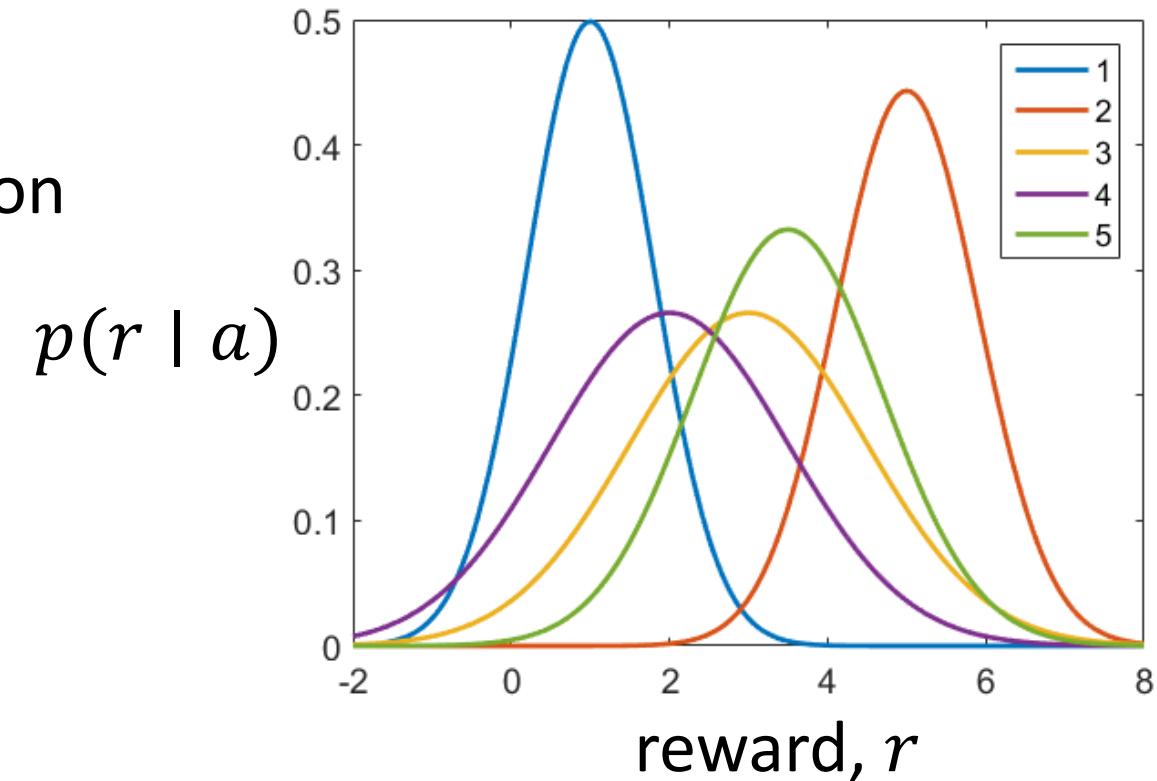
$$p(r \mid \pi) \triangleq \sum_a p(r \mid a)\pi(a)$$

- To simplify notations, we often write the objective function in this lecture

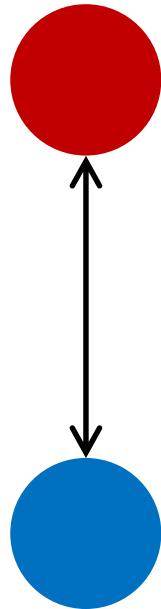
$$J(\pi) = \sum_{r,a} rp(r \mid a)\pi(a)$$

# Example

- The rewards of the machine are determined by a Gaussian distribution where
  - mean [1, 5, 3, 2, 3.5]
  - standard deviation [0.8, 0.9, 1.5, 1.5, 1.2]
- The optimal action is to select the machine “2” because it has the highest mean value
- A learning agent does not know these distributions



# Two RL Approaches



Value-based approach

- Compute expected rewards for every action
- Select an action according to the expected rewards

Policy-based approach

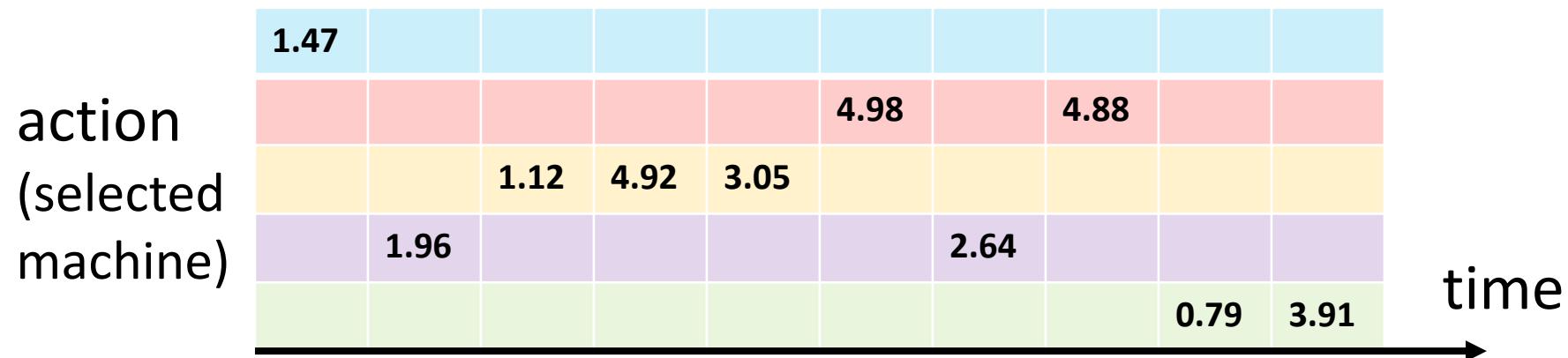
- Compute a gradient of an expected reward under a parametric stochastic policy
- Update a policy by a stochastic gradient ascent

# Value-based Approach

- We want to know the expected values of actions

$$q_*(a) \triangleq \mathbb{E}_\pi[R_t \mid A_t = a] = \int r p(r \mid a) dr$$

- Can't compute it because  $p(r \mid a)$  is unknown
- In order to estimate  $q_*(a)$ , we need to gather samples by playing
- Suppose that an agent gets the following sequence of rewards based on  $\pi(a) = 1/5$  for all  $a$



# Value-based Approach

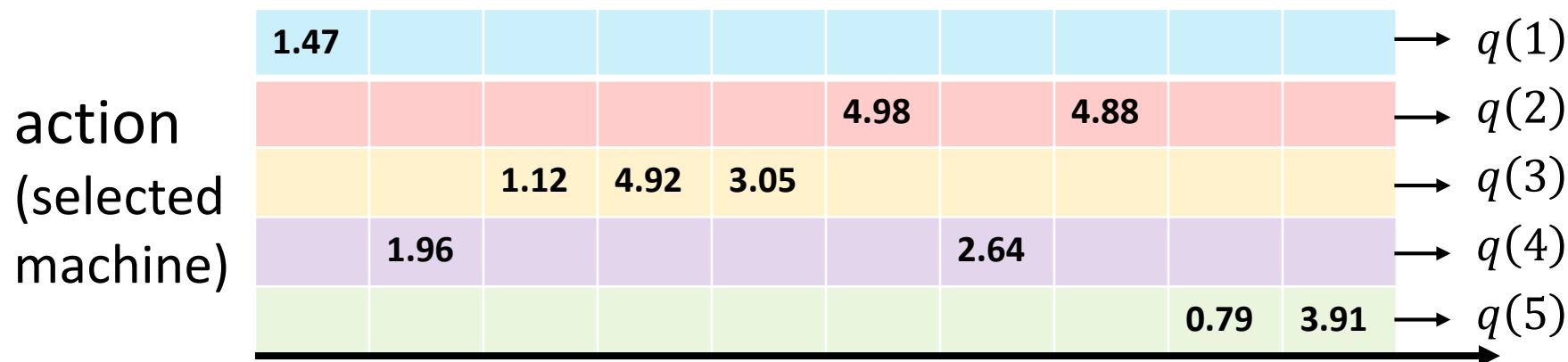
- Estimate by averaging the rewards actually received:

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

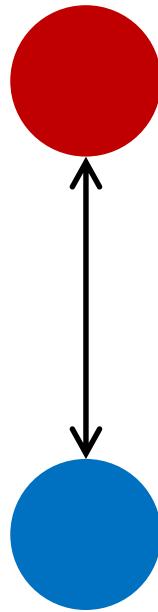
where  $Q_t(a)$  is an array estimate of  $q(a)$  at time  $t$

- Example

$$Q_5(3) = \frac{1.12 + 4.92 + 3.05}{3} = 3.03$$



# Action Selection Strategies during Learning

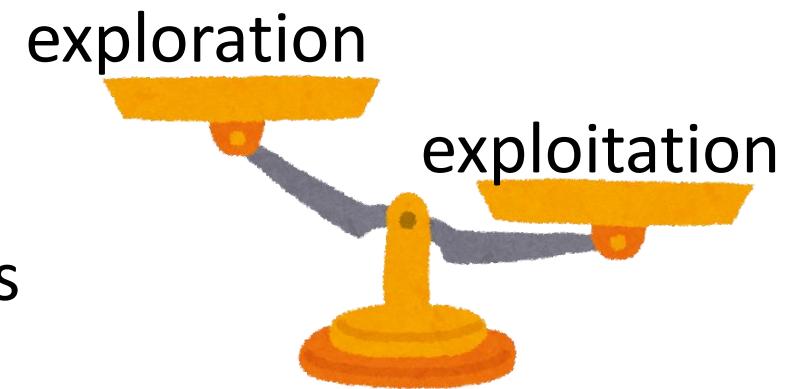


## Exploitation

- Use known information to act
- E.g., go to the best restaurant I know

## Exploration

- Find more information about machines
  - E.g., try a new restaurant
- 
- An RL agent faces the exploration-exploitation dilemma at every time step



# Greedy Policy

- If  $Q_t$  is sufficiently accurate, it is enough to select an action greedily
  - always choose the action with current best expected reward

$$A_t = \operatorname{argmax}_a Q_t(a)$$

- However, a learning agent does not have a correct  $Q$  at the beginning of learning
- So, the agent should gather information **by trial and error** to estimate  $Q$  efficiently
- How should the agent select actions during learning?

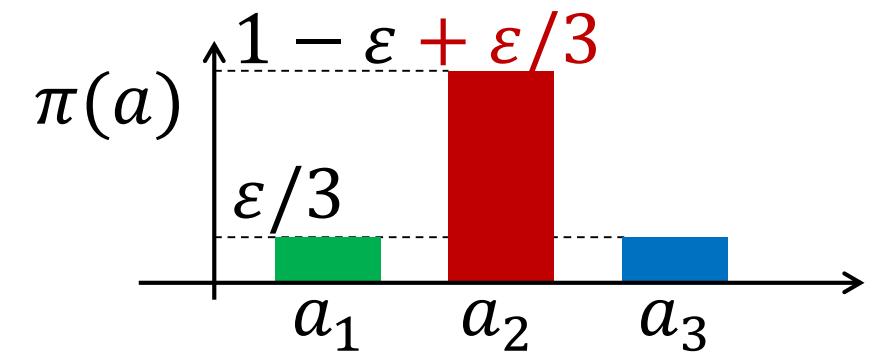
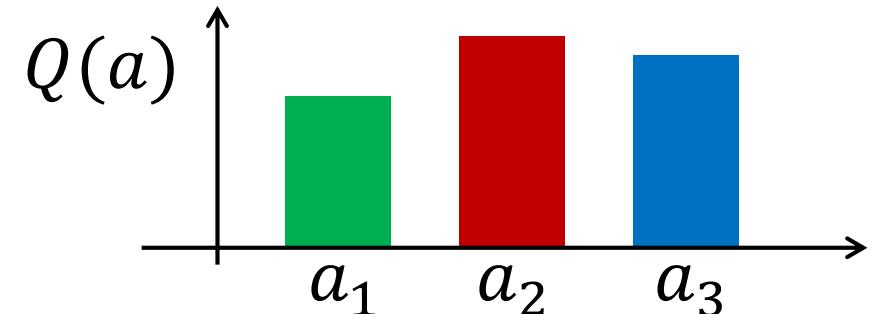


# $\varepsilon$ -greedy Policy

- Choose the action with current best expected reward with probability  $1 - \varepsilon$
- Choose another action randomly with probability  $\varepsilon/k$

$$\pi(a) = \begin{cases} \varepsilon/k + 1 - \varepsilon & \text{if } a = \underset{a'}{\operatorname{argmax}} Q(a') \\ \varepsilon/k & \text{otherwise} \end{cases}$$

- Exploration insensitive to relative performance levels



$\varepsilon = 1$   $\xrightarrow{\text{time}}$   $\varepsilon = 0$   
(exploration) (exploitation)

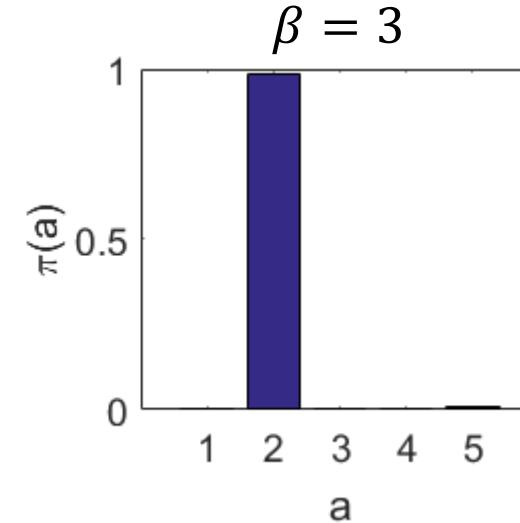
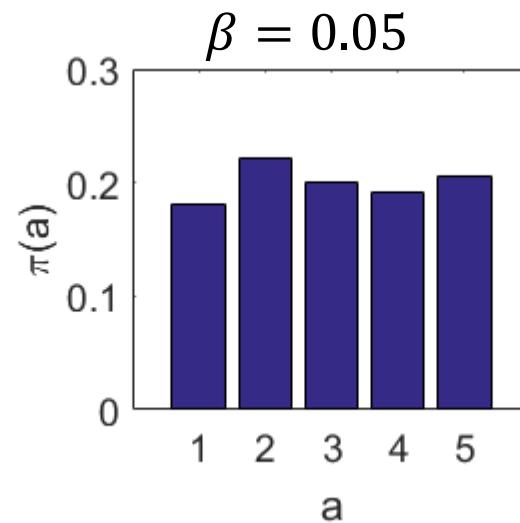
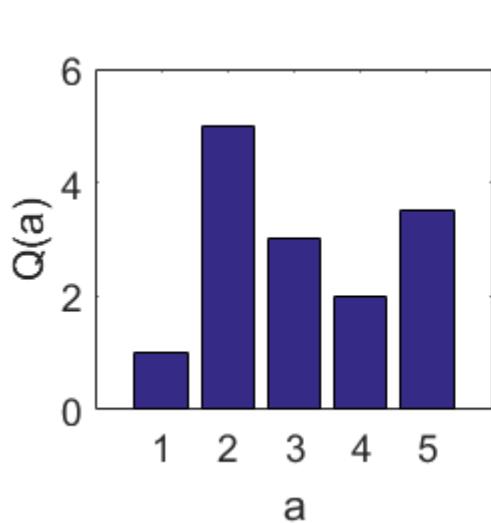
# Boltzmann Distribution

- The policy is calculated from the value

$$\pi(a) = \frac{\exp(\beta Q(a))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q(a'))}$$

where  $\beta$  is an inverse temperature

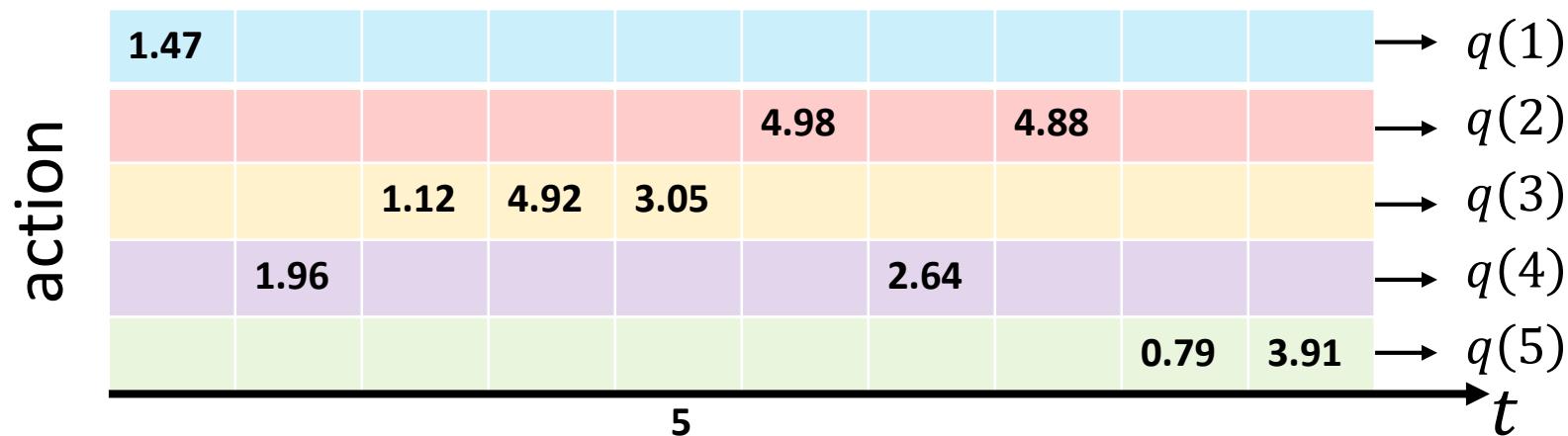
- $\beta = 0$ : pure exploration (random policy)
- $\beta \rightarrow \infty$ : pure exploitation (greedy policy)



$\beta = 0$  ————— time —————  $\beta = \infty$   
(exploration) (exploitation)

# Estimation of $Q$

- Let  $R^n(a)$  and  $Q^n(a)$  denote the reward received after the  $n$ -th selection of  $a$  and the estimate of its action value after it has been selected  $n - 1$  times



- $R^1(3) = 1.12, R^2(3) = 4.92, R^3(3) = 3.05$
  - $Q^1(3) \triangleq R^1(3) = 1.12$
  - $Q^2(3) = (R^1(3) + R^2(3))/2 = 3.02$
  - $Q^3(3) = (R^1(3) + R^2(3) + R^3(3))/3 = 3.03$

# Incremental Estimation of $Q$

- It is easy to devise incremental update rule to compute  $Q$

$$\begin{aligned} \bullet \quad Q^{n+1}(a) &= \frac{1}{n} \sum_{i=1}^n R^i(a) = \frac{1}{n} \left( R^n(a) + \sum_{i=1}^{n-1} R^i(a) \right) \\ &= \frac{1}{n} \left( R^n(a) + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} R^i(a) \right) \\ &= \frac{1}{n} (R^n(a) + (n-1)Q^n(a)) \\ &= Q^n(a) + \frac{1}{n} [R^n(a) - Q^n(a)] \end{aligned}$$

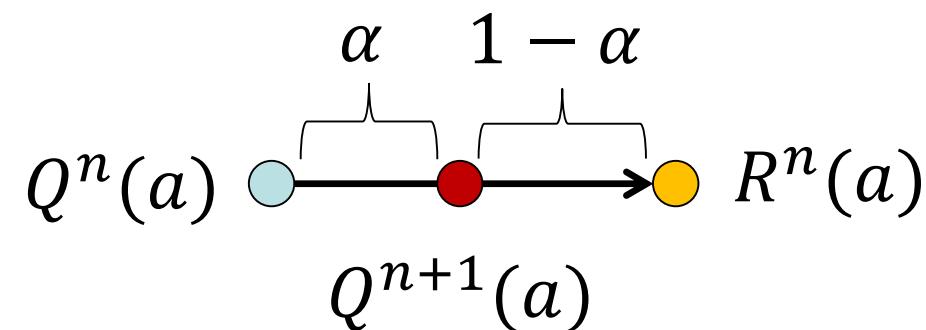
# Interpretation of the Update Rule

- $$Q^{n+1}(a) = Q^n(a) + \frac{1}{n} [R^n(a) - Q^n(a)]$$

↑  
new estimate                      ↑ target              ↑ old estimate

- $\delta \triangleq R^n(a) - Q^n(a)$  represents the error for the old estimate
- $\alpha \triangleq 1/n$  is a learning rate
- Online update rule for estimating rewards

$$Q^{n+1}(a) = (1 - \alpha)Q^n(a) + \alpha R^n(a)$$



# A Simple Bandit Algorithm

Initialize, for  $a = 1$  to  $k$ :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \arg \max_a Q(a) & \text{with probability } 1 - \varepsilon \text{ (breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

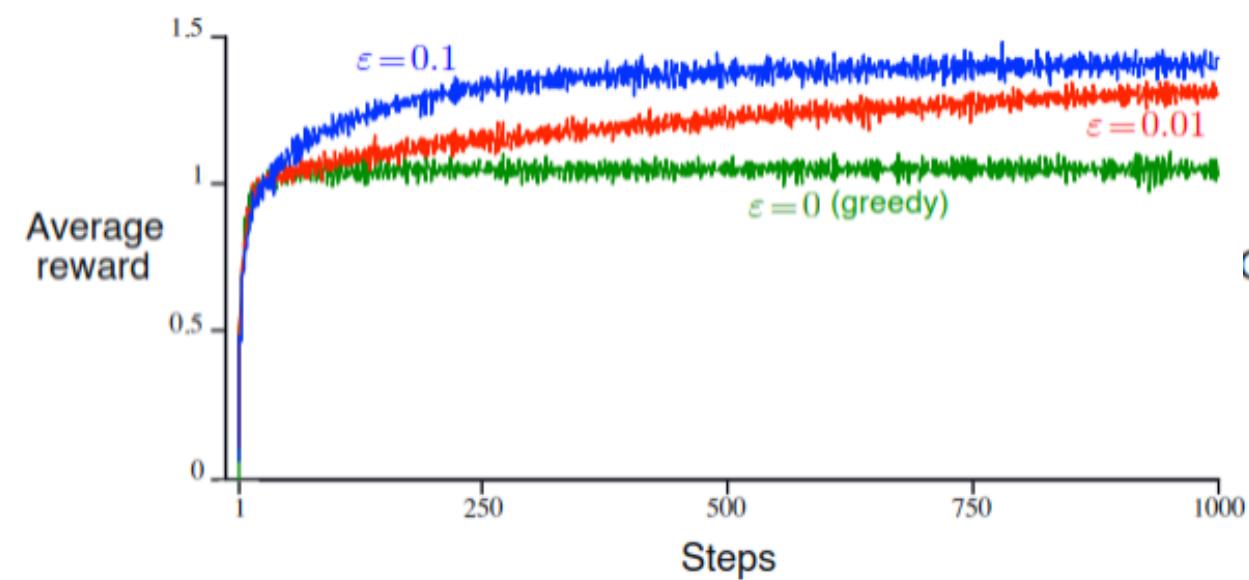
$$R \leftarrow \text{bandit}(A)$$

$$N(A) \leftarrow N(A) + 1$$

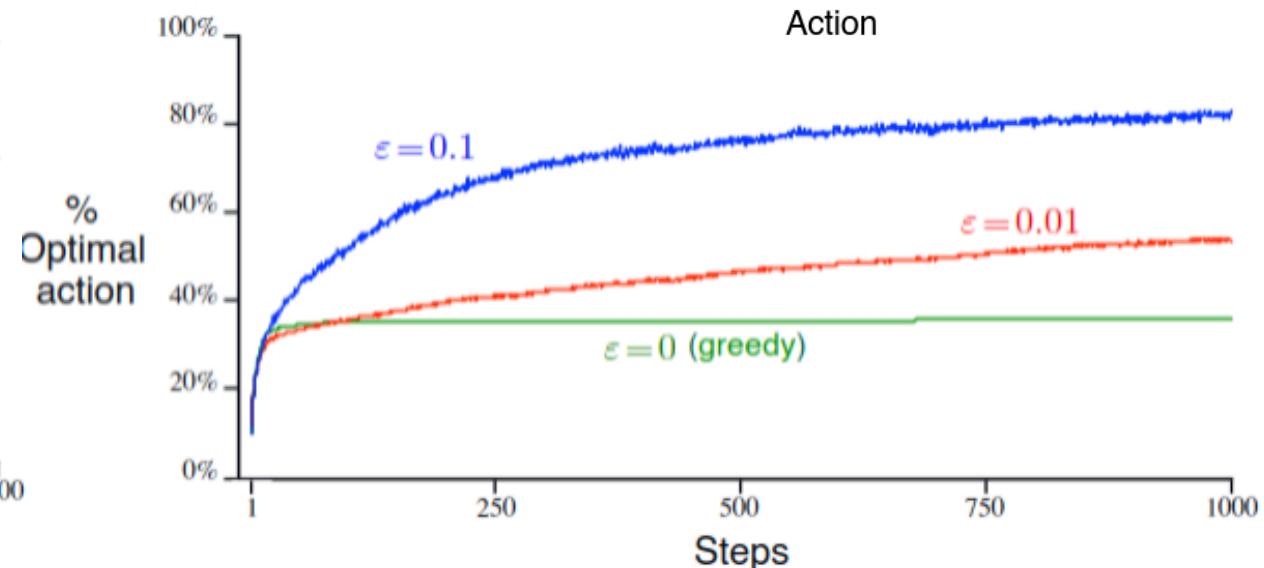
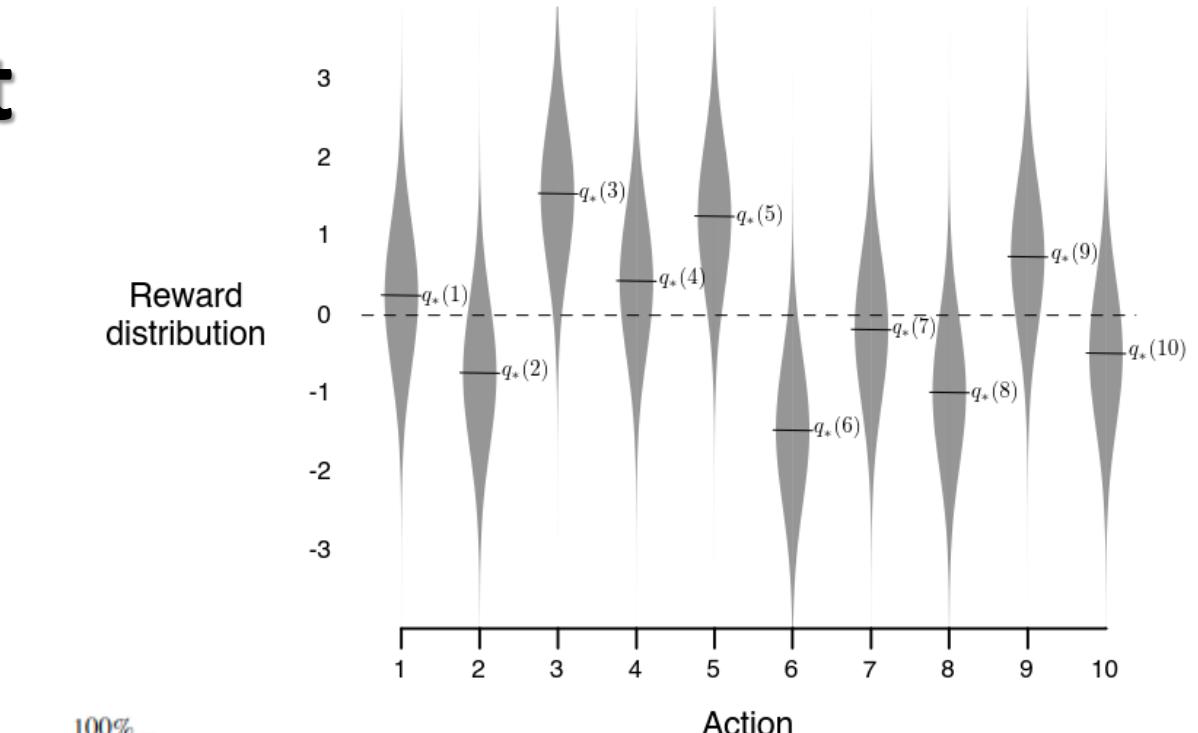
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

# Example: 10-armed Bandit

- Average performance of  $\varepsilon$ -greedy action-value method



Reward distribution

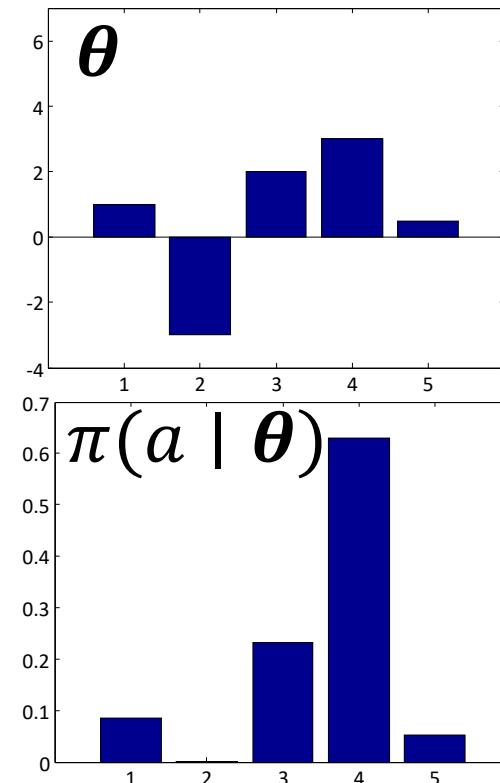


# Policy-based Approach

- So far, we explicitly estimate action values from experiences, and they are used to derive a policy
- Next, we consider how to learn the policy directly from experiences
- One way is to parameterize the policy by the Boltzmann distribution
- The Boltzmann policy is given by

$$\pi(a_i | \theta) = \frac{\exp(\theta_i)}{\sum_{j=1}^k \exp(\theta_j)}$$

where  $\theta = [\theta_1, \theta_2, \dots, \theta_k]^\top$



# Policy-based Approach

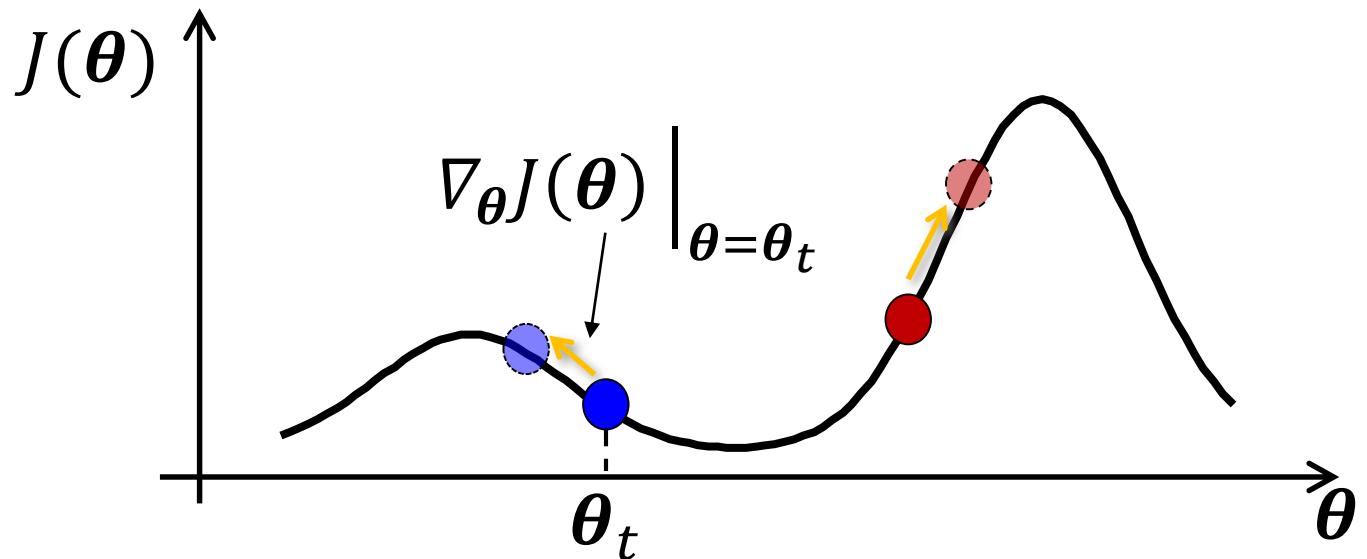
- The expected reward is

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[R] = \sum_{r,a} rp(r | a) \pi(a | \boldsymbol{\theta})$$

- The goal is to find  $\boldsymbol{\theta}$  that maximizes  $J(\boldsymbol{\theta})$
- Use a gradient ascent method to maximize  $J(\boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

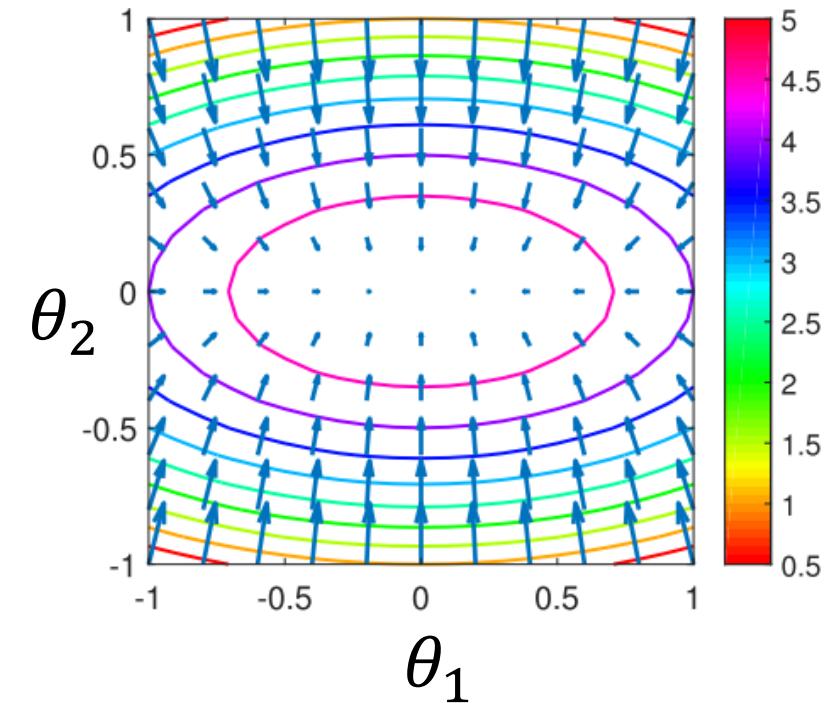
where  $\alpha$  is a positive step-size parameter



# Gradient of a Function

- $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_k]^\top$ :  $k$ -dimensional vector
- $J(\boldsymbol{\theta}) : \mathbb{R}^k \rightarrow \mathbb{R}$ : differentiable scalar function
- $\nabla_{\boldsymbol{\theta}} J : \mathbb{R}^k \rightarrow \mathbb{R}^k$ : its gradient

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} J(\theta_1, \dots, \theta_k) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_k} \end{bmatrix} = \frac{\partial J}{\partial \boldsymbol{\theta}}$$



$$J(\boldsymbol{\theta}) = -\theta_1^2 - 4\theta_2^2 + 5$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} -2\theta_1 \\ -8\theta_2 \end{bmatrix}$$

# Gradient Ascent

- Find a local optimal solution

Initialize

$n \leftarrow 0$

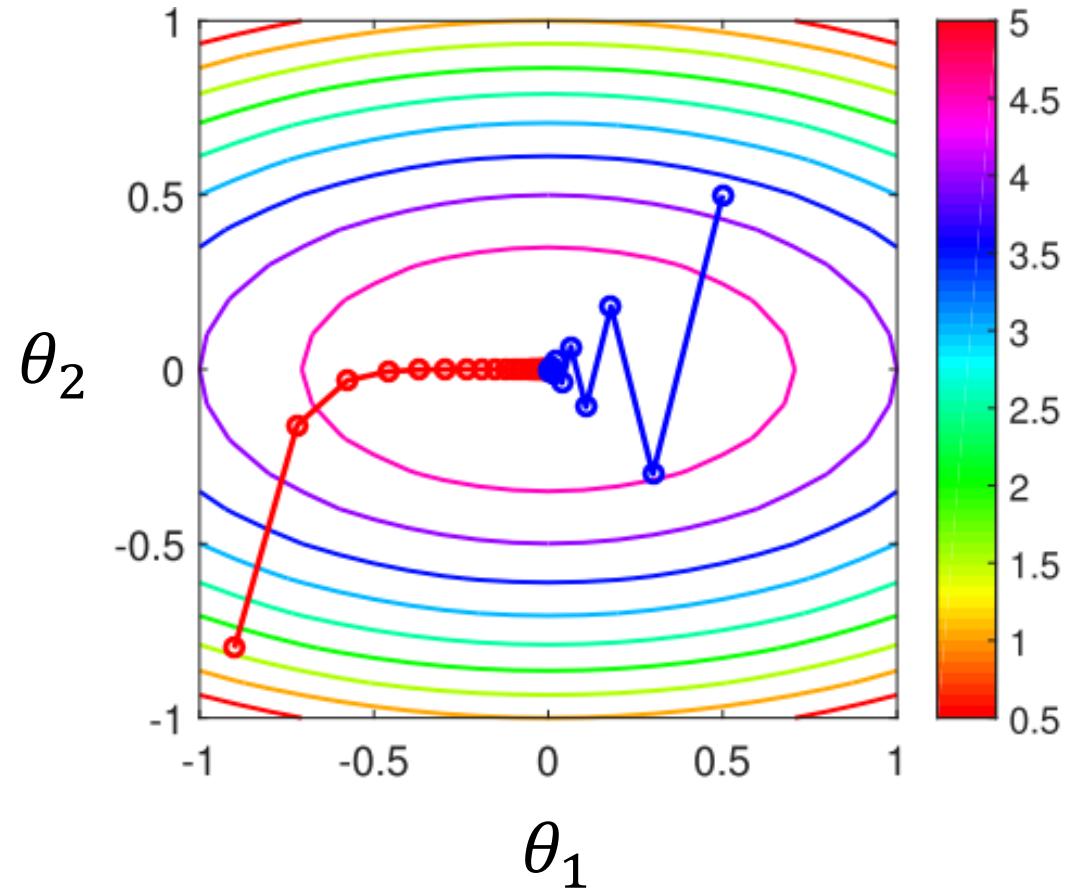
Repeat:

$n \leftarrow n + 1$

$\theta^n \leftarrow \theta^{n-1} + \alpha \nabla_{\theta} J(\theta)$

until stopping\_criterion( $\theta^n, \theta^{n-1}, \epsilon$ )

return  $\theta^n$



# Calculate $\nabla_{\theta}J(\theta)$

- $$\begin{aligned}\nabla_{\theta}J(\theta) &= \frac{\partial}{\partial \theta} \left[ \sum_{r,a} rp(r | a) \pi(a | \theta) \right] \\ &= \sum_{r,a} rp(r | a) \frac{\partial \pi(a | \theta)}{\partial \theta} \\ &= \sum_{r,a} rp(r | a) \pi(a | \theta) \frac{\partial \ln \pi(a | \theta)}{\partial \theta} \\ &= \mathbb{E}_{\pi} \left[ R \frac{\partial \ln \pi(a | \theta)}{\partial \theta} \right]\end{aligned}$$

–  $\ln x$  is the natural logarithm of  $x$

## Calculate $\partial \ln \pi(a_i | \theta) / \partial \theta$ (1/2)

- $\pi(a_i | \theta) = \frac{\exp(\theta_i)}{\sum_{j=0}^{k-1} \exp(\theta_j)}$
- $$\begin{aligned}\frac{\partial \pi(a_i | \theta)}{\partial \theta_i} &= \frac{\exp(\theta_i) \sum_{j=0}^{k-1} \exp(\theta_j) - (\exp(\theta_i))^2}{\left( \sum_{j=0}^{k-1} \exp(\theta_j) \right)^2} \\ &= \frac{\exp(\theta_i)}{\sum_{j=0}^{k-1} \exp(\theta_j)} - \left( \frac{\exp(\theta_i)}{\sum_{j=0}^{k-1} \exp(\theta_j)} \right)^2 = \pi(a_i | \theta)(1 - \pi(a_i | \theta))\end{aligned}$$
- $$\frac{\partial \ln \pi(a_i | \theta)}{\partial \theta_i} = \frac{1}{\pi(a_i | \theta)} \frac{\partial \pi(a_i | \theta)}{\partial \theta_i} = 1 - \pi(a_i | \theta)$$

## Calculate $\partial \ln \pi(a_i | \theta) / \partial \theta$ (2/2)

- $\frac{\partial \pi(a_i | \theta)}{\partial \theta_j} = -\frac{\exp(\theta_i) \exp(\theta_j)}{\left(\sum_{j=0}^{k-1} \exp(\theta_j)\right)^2} = -\pi(a_i | \theta) \pi(a_j | \theta)$
- $\frac{\partial \ln \pi(a_i | \theta)}{\partial \theta_j} = \frac{1}{\pi(a_i | \theta)} \frac{\partial \pi(a_i | \theta)}{\partial \theta_j} = -\pi(a_j | \theta)$

# Implementation

- Compute the gradient from experiences

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi}[R \nabla_{\theta} \ln \pi(a | \theta)]$$

- Use the stochastic gradient method
  1. Sample action:  $A \sim \pi(a | \theta)$
  2. Receive the reward  $R$  from the environment:  
 $R \sim p(r | A)$
  3. Update  $\theta$  by the stochastic gradient ascent

$$\theta \leftarrow \theta + \alpha R \frac{\partial \ln \pi(A | \theta)}{\partial \theta}$$

# Variance Reduction

- Although we can estimate policy gradient, it has a large variance in general
- We can further reduce the variance by subtracting a baseline

$$\begin{aligned} & \mathbb{E}_\pi \left[ (R - b) \frac{\partial \ln \pi(A | \theta)}{\partial \theta} \right] \\ &= \mathbb{E}_\pi \left[ R \frac{\partial \ln \pi(A | \theta)}{\partial \theta} \right] + b \mathbb{E}_\pi \left[ \underbrace{\frac{\partial \ln \pi(A | \theta)}{\partial \theta}}_{=0} \right] \end{aligned}$$

# Proof Sketch

- We want to prove  $\mathbb{E}_\pi \left[ \frac{\partial \ln \pi(A | \theta)}{\partial \theta} \right] = 0$
- $$\begin{aligned} \mathbb{E}_\pi \left[ \frac{\partial \ln \pi(A | \theta)}{\partial \theta} \right] &= \sum_{r,a} p(r | a) \pi(a | \theta) \frac{\partial \ln \pi(a | \theta)}{\partial \theta} \\ &= \sum_{r,a} p(r | a) \frac{\partial \pi(a | \theta)}{\partial \theta} = \frac{\partial}{\partial \theta} \left[ \sum_{r,a} p(r | a) \pi(a | \theta) \right] \\ &= \frac{\partial}{\partial \theta} [1] = 0 \end{aligned}$$
- This fact suggests that the policy gradient remains unchanged even though we subtract a baseline constant

# Summary

- Introduction to reinforcement learning
- Bandit problem
  - Action, Reward, Value
- Value-based approach: Estimate the value function and derive the policy from the estimated value function
  - Indirect approach
- Policy-based approach: Estimate the gradient of the objective function with respect to the policy parameters and update the parameters
  - Direct approach

# References

- Abe, N., et al. Optimizing debt collections using constrained reinforcement learning. In Proc. of the 16th ACM SIGKDD, pp. 75-84, 2010.
- Abbeel, P., Coates, A., and Ng, A.Y. [Autonomous helicopter aerobatics through apprenticeship learning](#). International Journal of Robotics Research, 2010.
- Kober, J., et al. [Reinforcement Learning in Robotics: A Survey](#). International Journal of Robotics Research, 32(11): 1238-1274, 2013.
- Miller, G., ..., Abe, N., et al. [Tax collection optimization for New York State](#). Interfaces, 42(1), 74-84, 2012.
- Mnih, V., et al. [Huma-level control through deep reinforcement learning](#). Nature, 518:529-533, 2015.
- Pathak, D., Agrawal, P., Efros, A.A., Darrell, T. Curiosity-driven Exploration by Self-supervised Prediction. In Proc. of ICML, 2017.

# References

- Silver, D., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489, 2016.
- Silver, D., et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676), 354–359, 2017.
- Shure, L. Multi-Armed Bandit Problem and Exploration vs. Exploitation Trade-off.  
<https://blogs.mathworks.com/loren/2016/10/10/multi-armed-bandit-problem-and-exploration-vs-exploitation-trade-off/>
- Sutton, R.S., and Barto, A.G. (2018). Reinforcement Learning: An Introduction (Second edition). The MIT Press.