Brain-Inspired Artificial Intelligence 1: Introduction to Reinforcement Learning

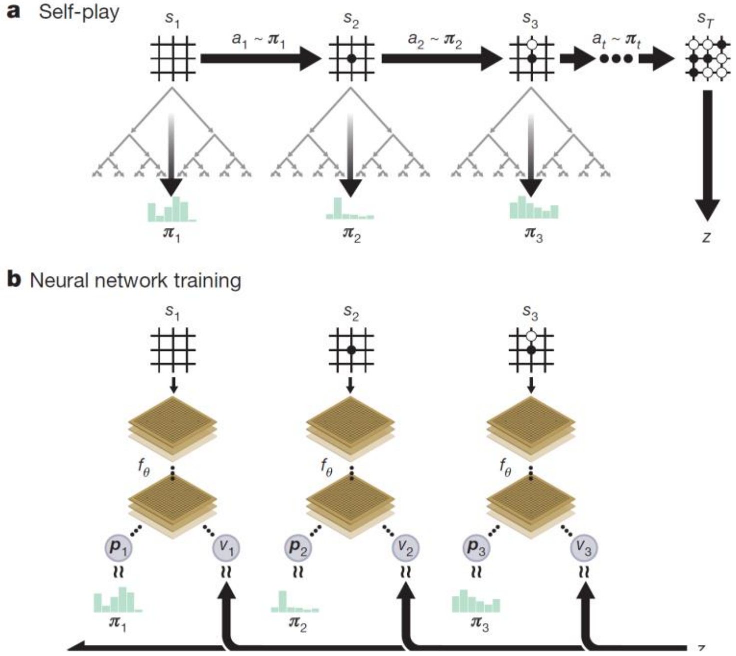
Eiji Uchibe

Dept. of Brain Robot Interface

ATR Computational Neuroscience Labs.

Reinforcement learning in games

AlphaGo Zero (Silver et al., 2017)
board game, Go
RL from scratch
4.9 millions of self-play

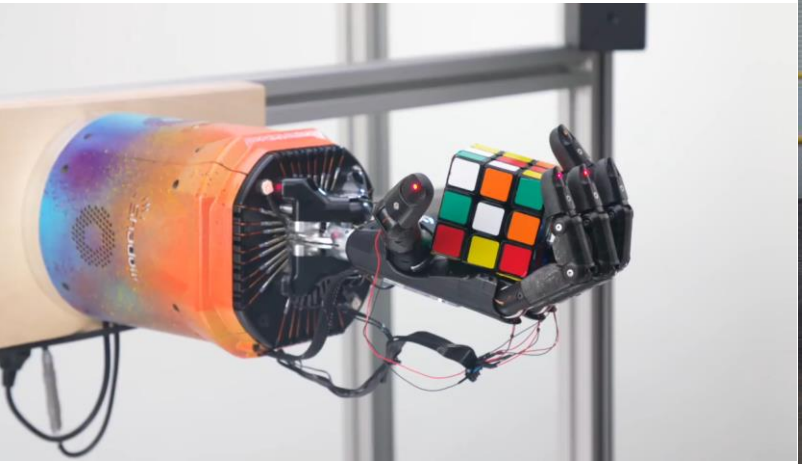


AlphaStar (Vinyals et al., 2019)
multiagent real-time strategy game
RL + supervised learning
200 years

Gran Turismo Sophy (Wurman et al., 2022)
racing game
RL with shaped rewards
1,000 PlayStation 4

Reinforcement learning in robotics

OpenAI (Akkaya et al., 2019)
manipulating Rubik's cube
RL + domain randomization
2.8 GWh of electricity

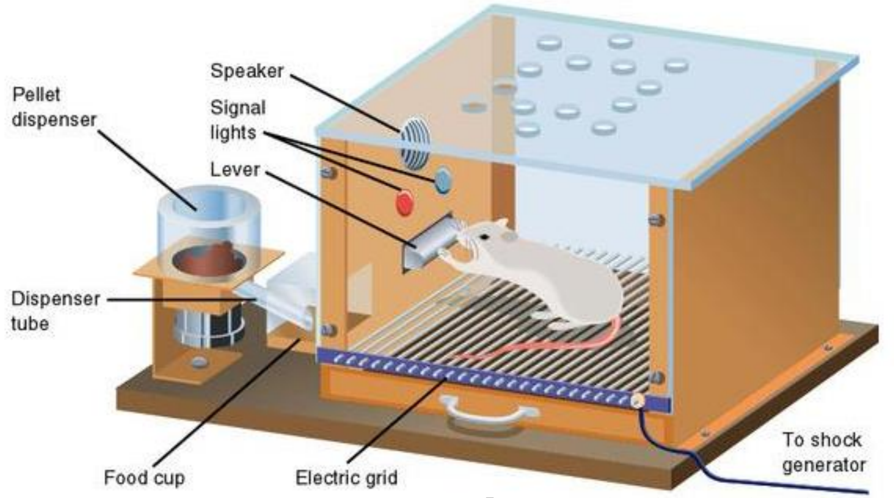


MT-Opt (Kalashnikov et al., 2021)
grasping
RL (+ supervised learning)
7 robots, 9600 robot hours

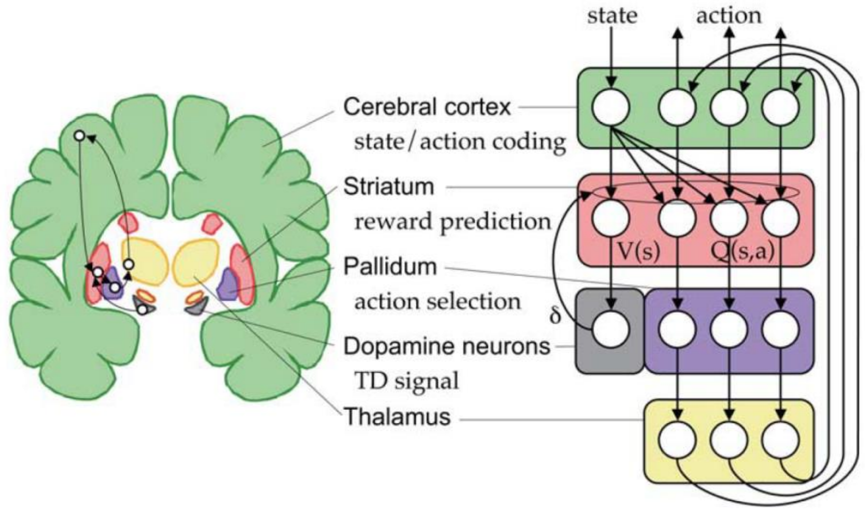
DDPP (Tsurumine et al., 2019)
folding a T-shirt
SL + RL with shaped rewards
192 demonstrations

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
- Computational model of decision making of human/animal

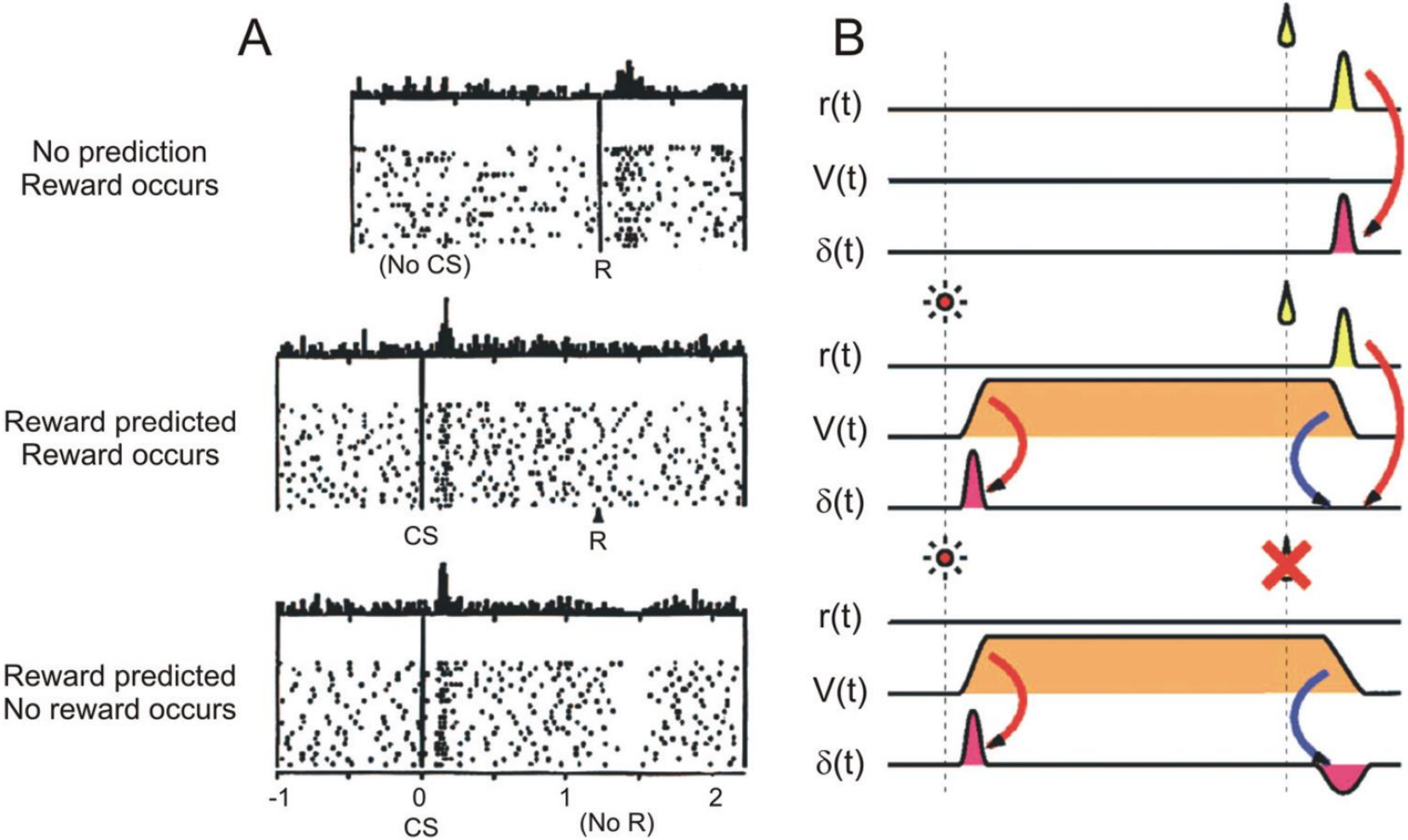


[Simply Psychology]



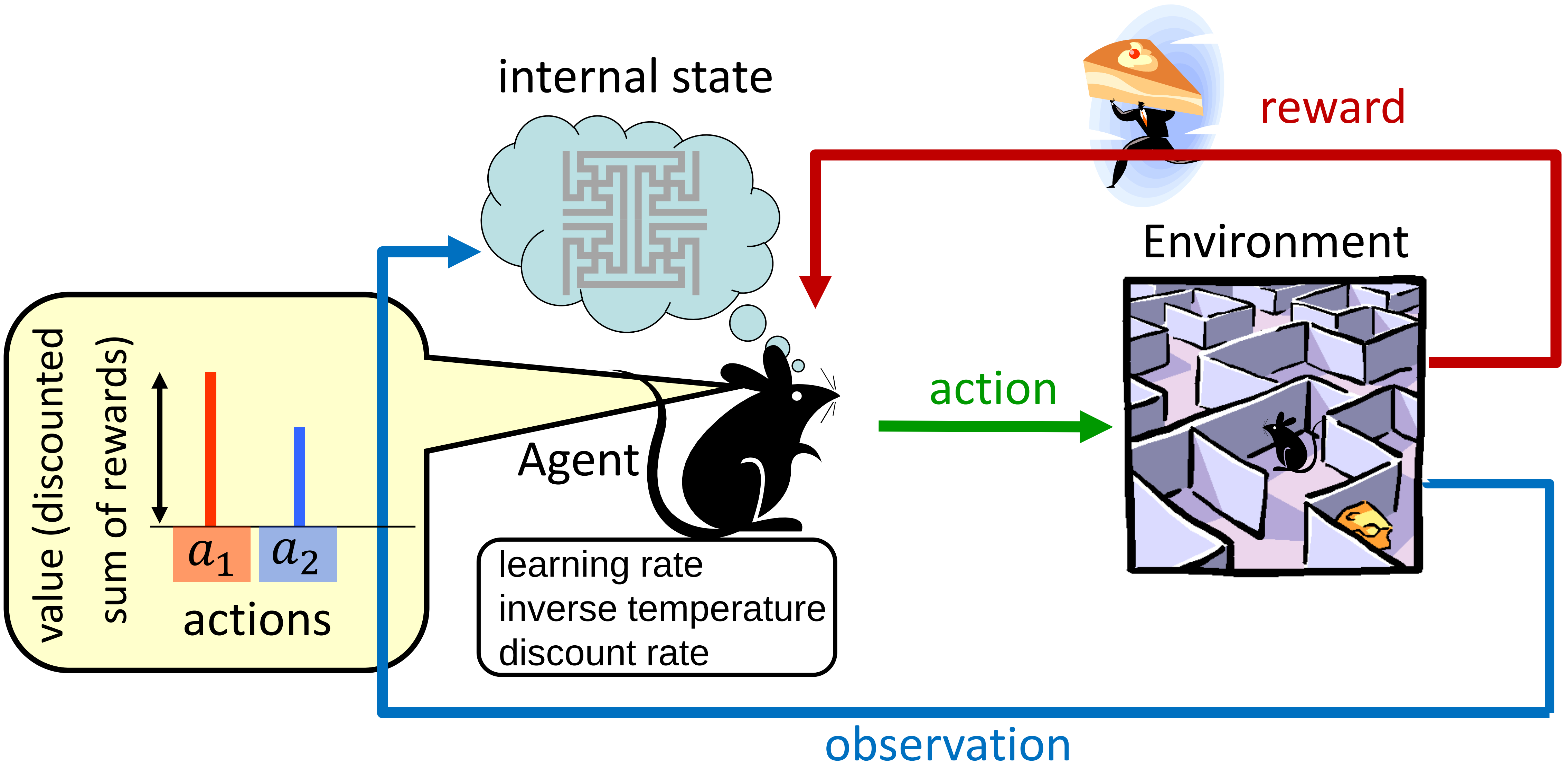
K. Doya (2007). Reinforcement learning: Computational theory and biological mechanisms. HFSP Journal, vol. 1, no. 1, pp. 30–40.

Dopamine neurons code Temporal Difference error



W.P. Schultz, P. Dayan, P.R. Montague (1997). A Neural Substrate of Prediction and Reward. Science 275, no. 5306: 1593–99.

Interaction between Agent and Environment



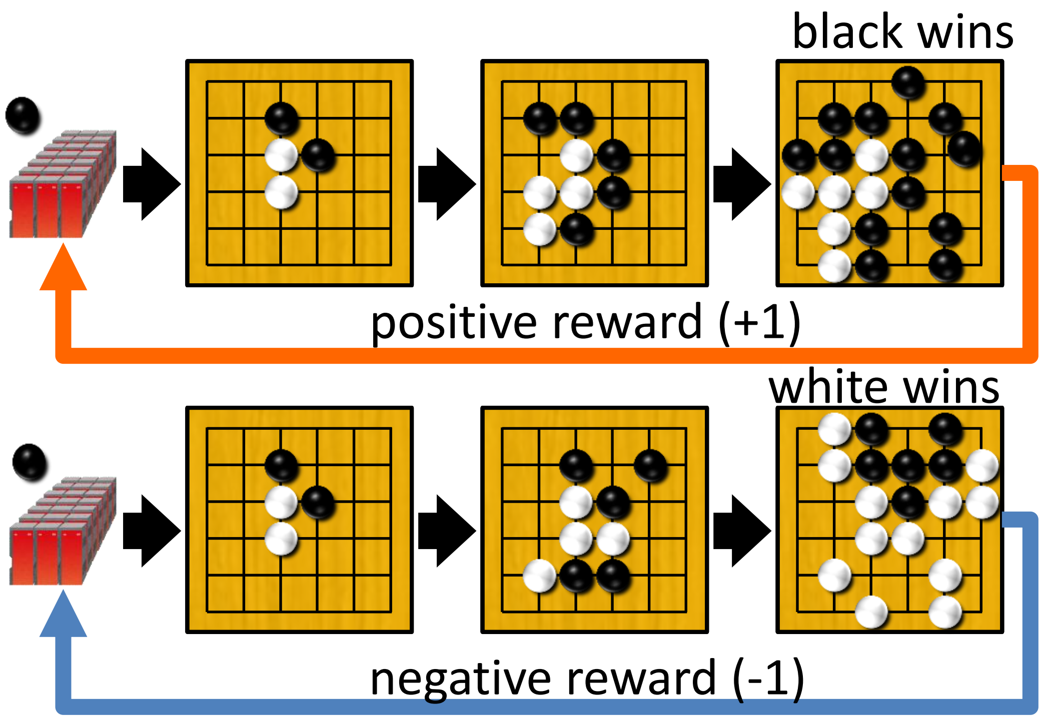
What is Reward?

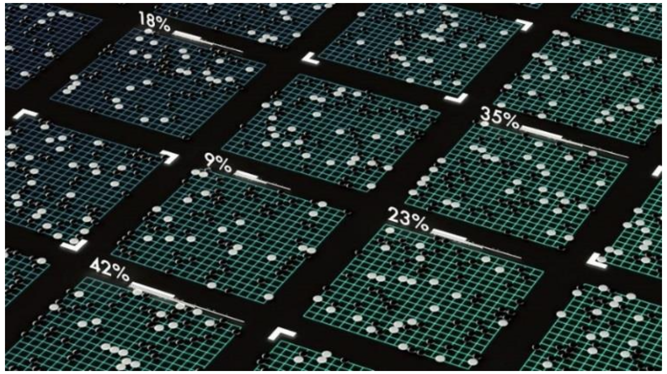
- In the case of Go
  - positive reward for winning
  - negative reward for losing
  - zero otherwise

➡ very sparse reward

⬇

- AlphaGo Zero, which does not use a record of a game of go, needs **4.9 million** games of self-play

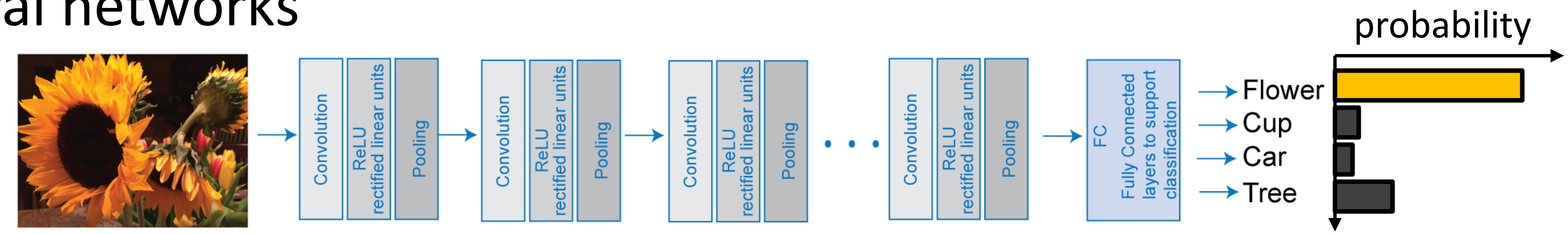




Difference between Deep Learning and (Deep) Reinforcement Learning

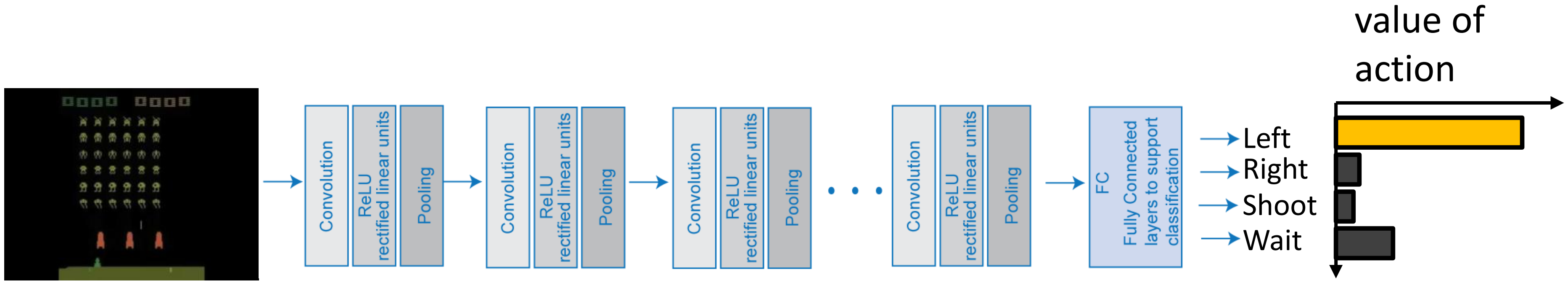
- Deep Learning for Classification

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



- Deep Reinforcement Learning

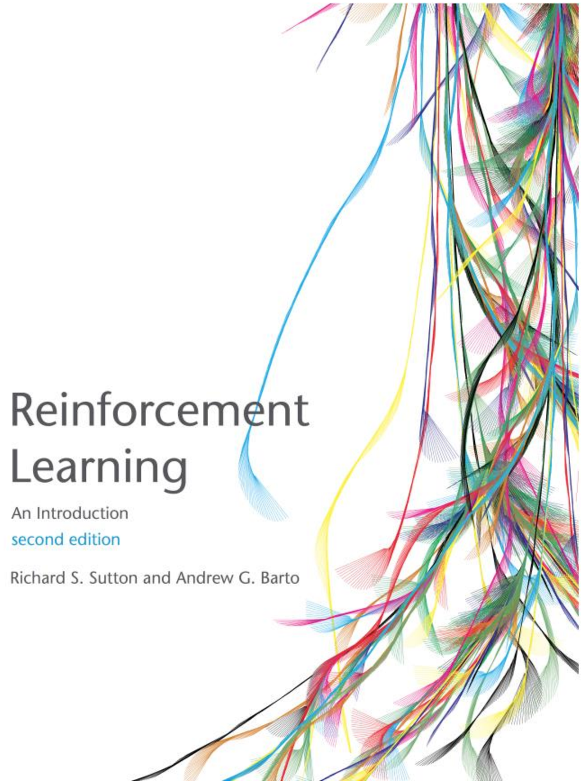
  Find a policy that maximizes the sum of rewards by trial and error



RECOMMENDED TEXTBOOKS

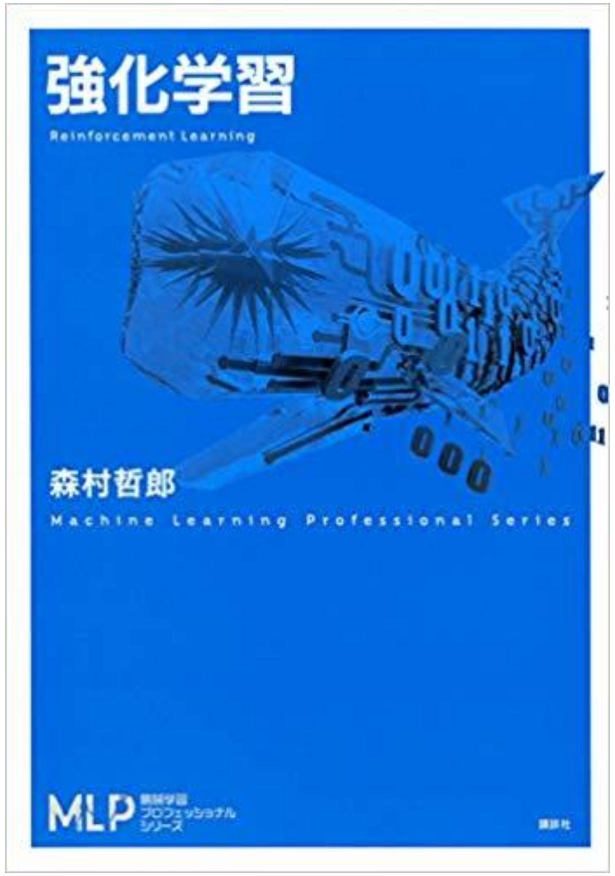
R.S. Sutton and A.G. Barto. (2018). Reinforcement learning: An introduction. MIT Press.

- First edition (1998), and Japanese translation (2000), and Second edition (2018).
- Must-read for researchers



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

- Official page
- Theoretical introduction to reinforcement learning with discrete states and actions
- One chapter describes recent topics such as distributional RL and deep RL



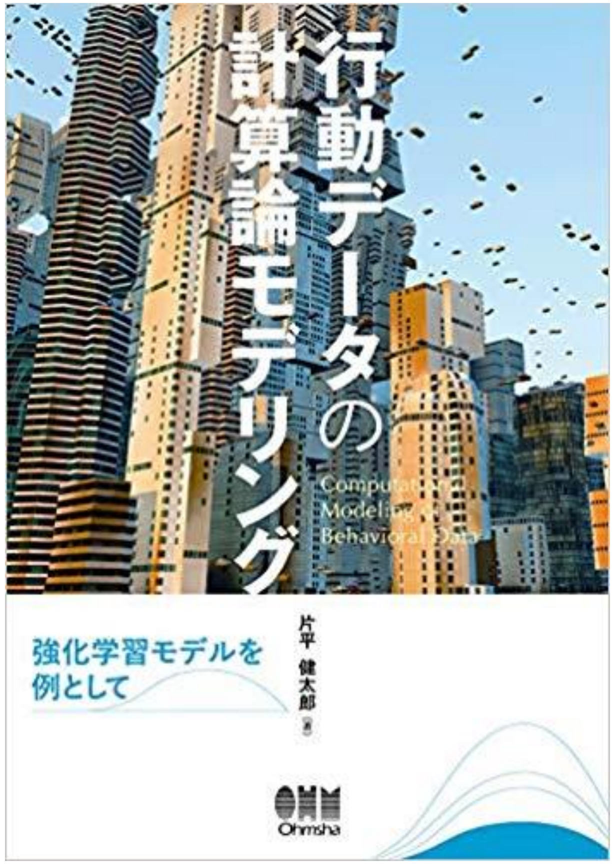
牧野貴樹、澁谷長史(編) (2016) これからの強化学習. 森北出版

- Official page

- Overview of modern reinforcement learning such as policy search methods, deep RL, and inverse RL

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

- Official page
- R and Stan scripts are available
- Computational modeling of behavioral data based on reinforcement learning



Other books

- 伊藤真 (2021). 「強化学習」を学びたい人が最初に読む本. 日経BP.
- 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.
- 斎藤康毅 (2022).ゼロから作るDeep Learning 4 —強化学習編.
  オライリージャパン.
- Szepesvári, C. (2010). Algorithms for reinforcement learning. Morgan & Claypool.
  - 翻訳: 小山田 創哲 訳者代表 (2017). 速習 強化学習アルゴリズム. 共立出版.

Report

- Please select one topic and write your report

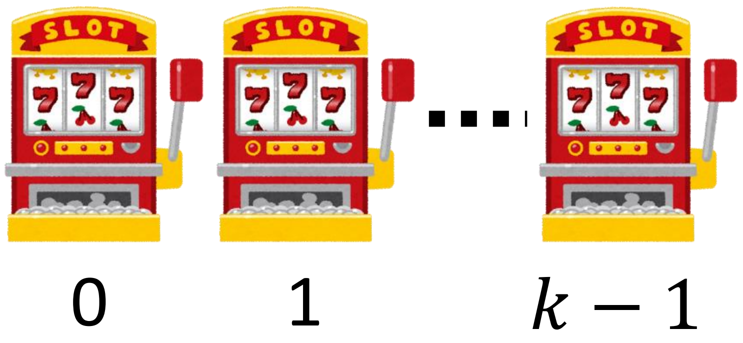
1. Consider the application of reinforcement learning

2. How can we reduce the training data and time?

REINFORCEMENT LEARNING FOR BANDIT PROBLEMS

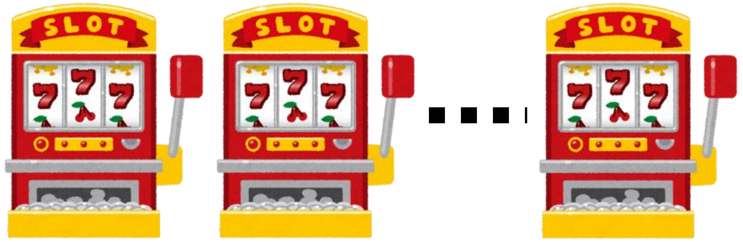
Multi-Armed Bandit Problem

- There exist k gambling machines
- Each machine has a different unknown probability distribution for rewards
  - Action: select a gambling machine
- The goal is to maximize the rewards obtained by successively playing gamble machines (the 'arms' of the bandits)



Components: Action and Policy

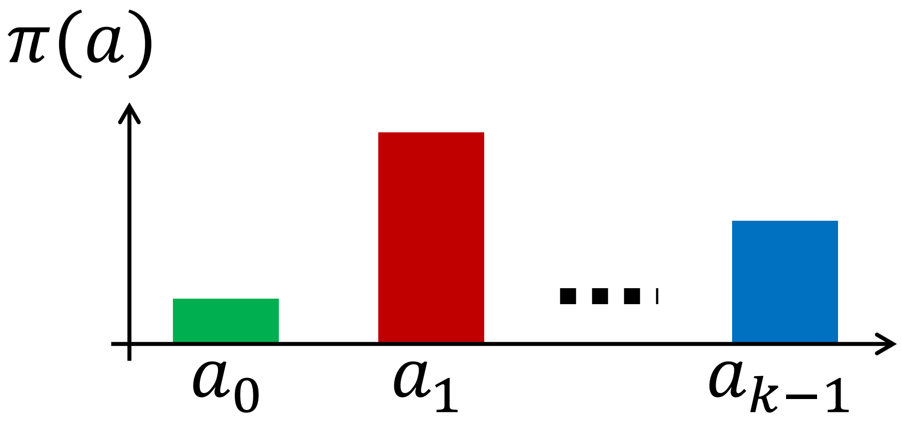
- (Discrete) Action
  - Choosing one gambling machine in the bandit problems
  - If there exist k machines, it is convenient to define a set of actions by $\mathcal{A} = \{a_0, a_1, \dots, a_{k-1}\}$



0 1 $k-1$

- Policy (learned by the agent)
  - $\pi(a)$: probability to select an action a
  - $\pi(a) \geq 0, \forall a \in \mathcal{A}$

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

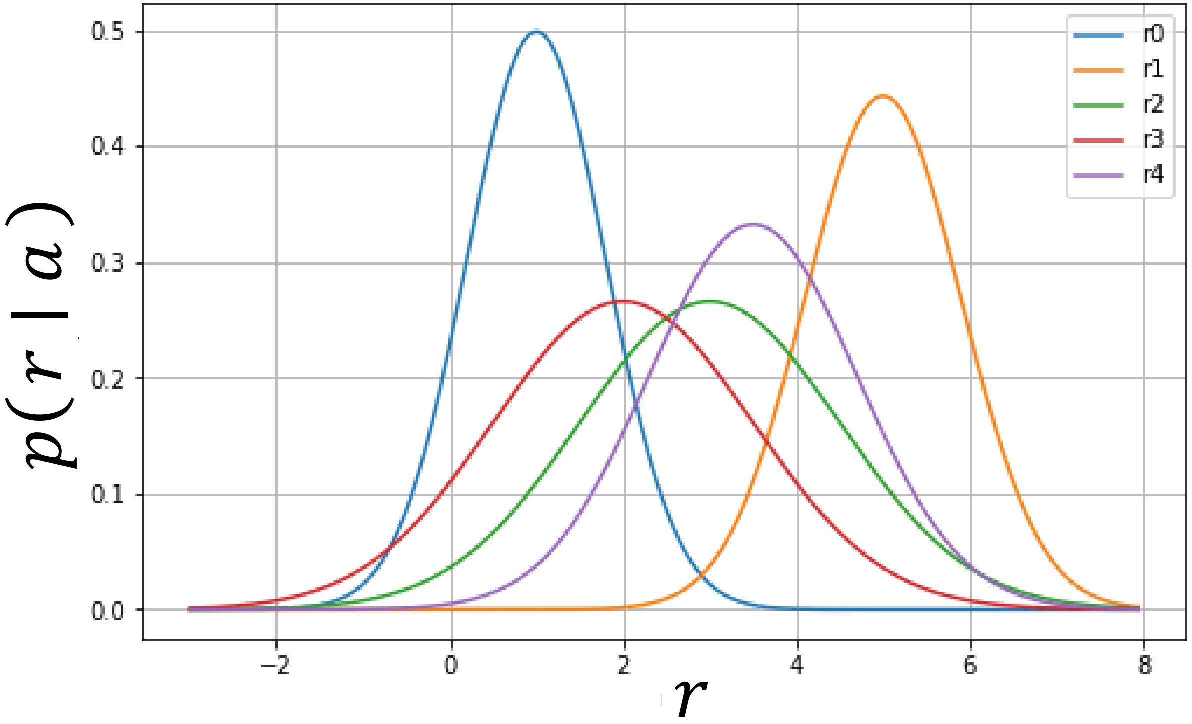


Components: Reward

- Usually unknown to a learning agent
  - When the agent executes an action a_t at time t, it receives a scalar feedback called reward

$$R_t \sim p(r \mid a_t)$$

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



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

Goal of the Learning Agent

- Find a policy π that maximizes the expected reward

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

- $p(r \mid \pi)$ 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 as follows in this lecture

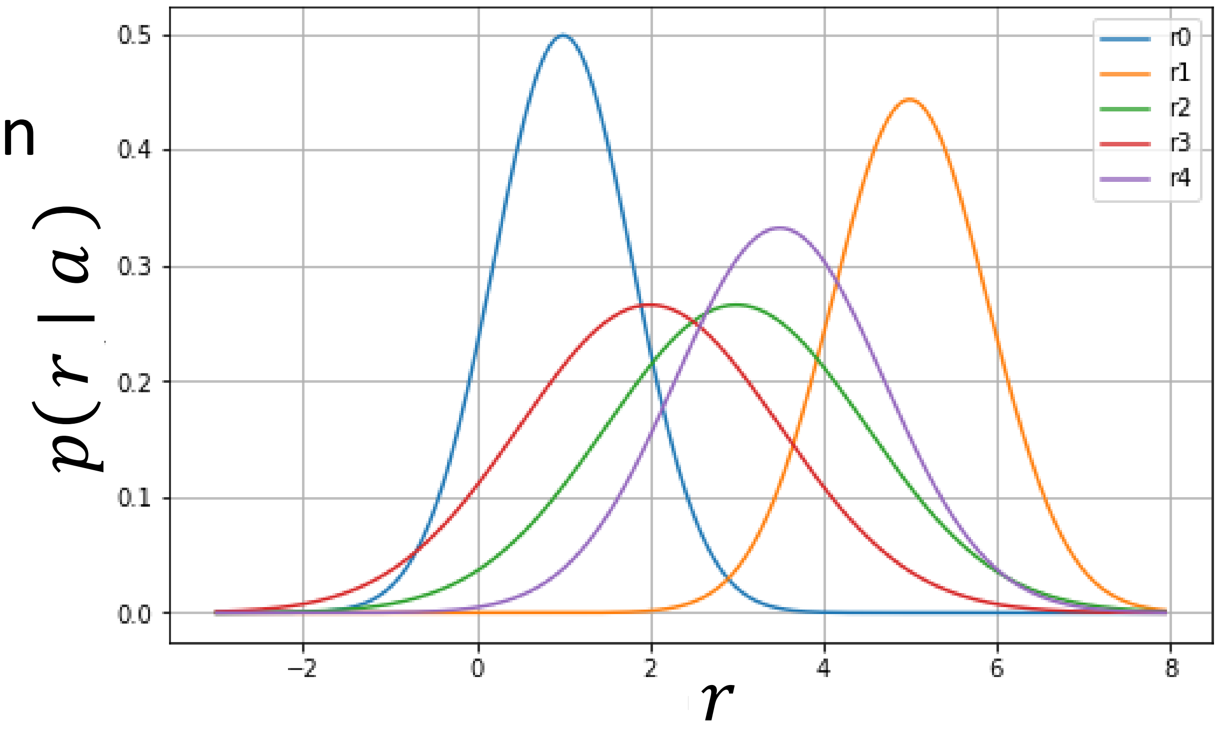
$$J(\pi) = \sum_{r,a} rp(r \mid a)\pi(a)$$

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 “1” because it has the highest mean value

- A learning agent does not know these distributions

REINFORCEMENT LEARNING FOR BANDIT PROBLEMS
-- VALUE-BASED APPROACH --

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

$$Q^*(a) \triangleq \mathbb{E}_{r\sim p(\cdot|a)}[r \mid a_t = a] = \int rp(r \mid a)\,\mathrm{d}r$$

  – Can't compute $Q^*(a)$ analytically 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

action (selected machine)

1.47									
					4.98		**4.88**		
		1.12	**4.92**	**3.05**					
	1.96					**2.64**			
								0.79	**3.91**

time

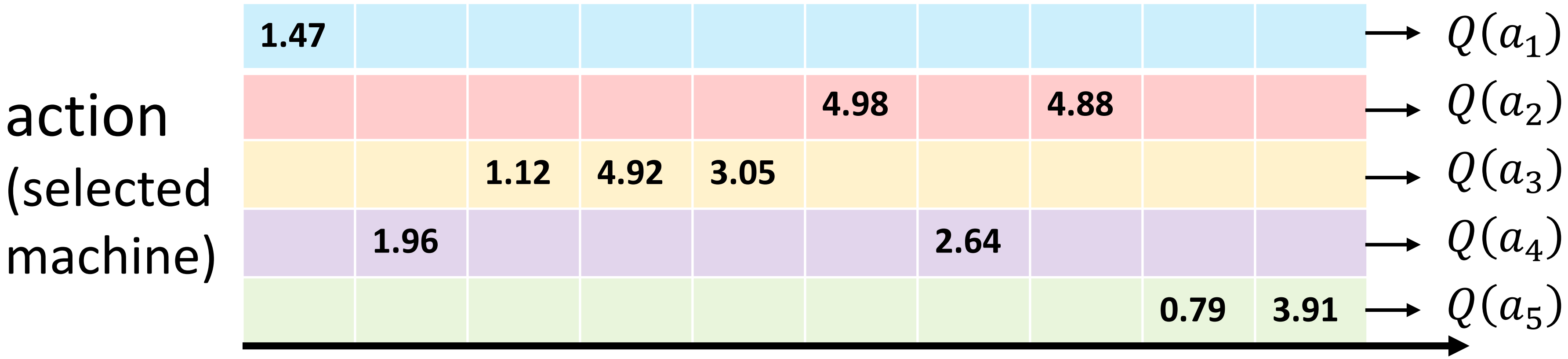
Value-based Approach

- 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 $$Q_4(a_2) = \frac{1.12 + 4.92 + 3.05}{3} = 3.03$$

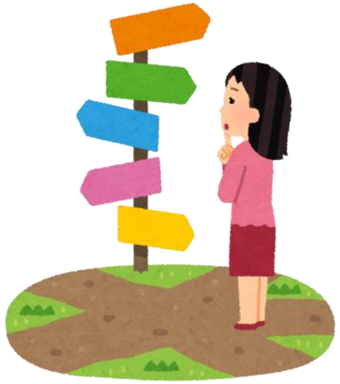


Greedy Policy

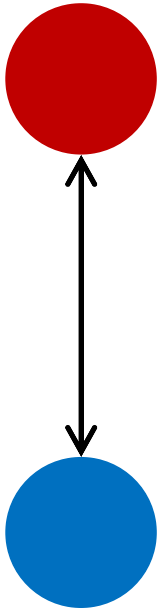
- If Q_t is sufficiently accurate, it is enough to select an action greedily
  - always choose the action with current best expected reward

$$a_t = \underset{a}{\operatorname{argmax}}\, Q_t(a)$$

- 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?



Action Selection Strategies during Learning

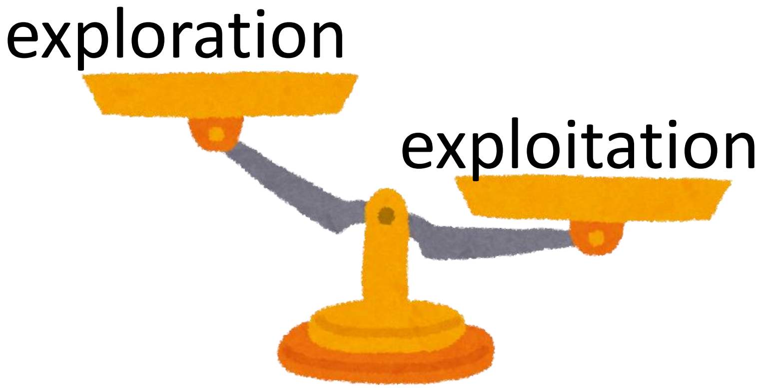


Exploitation

- 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

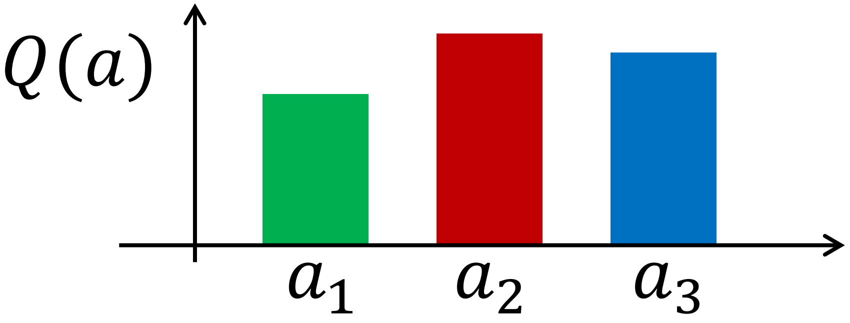
ε-greedy policy

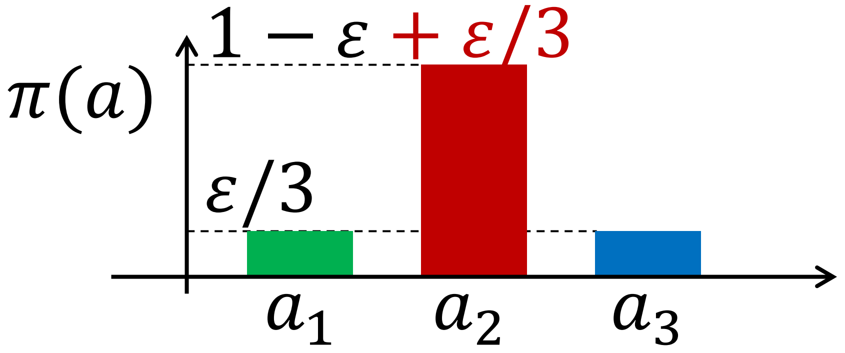
Open in Colab

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

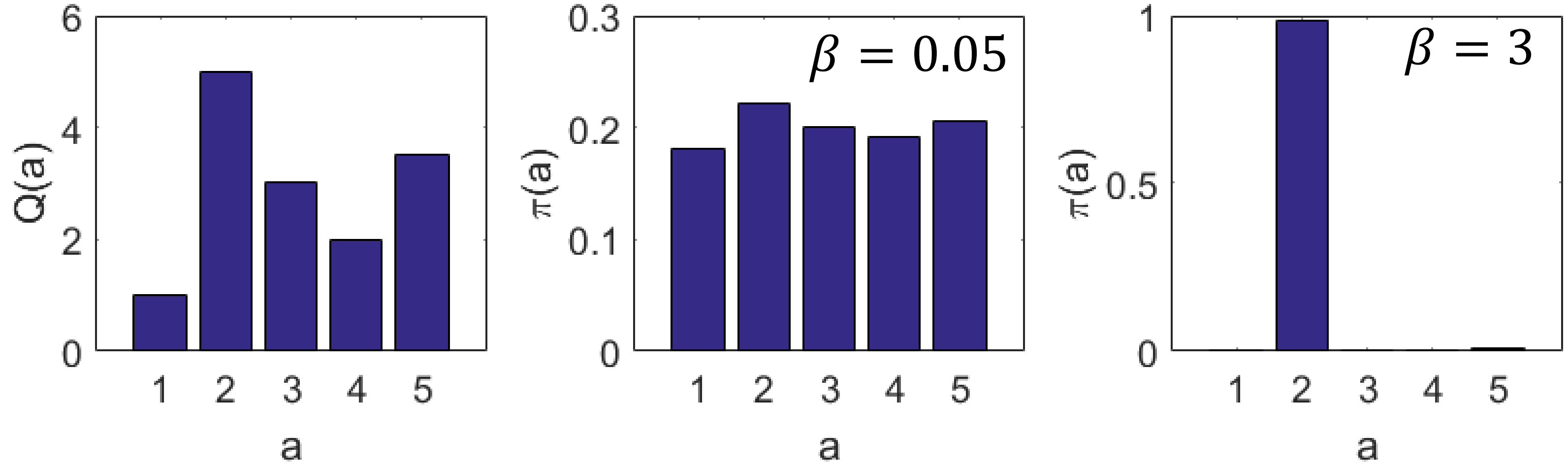
Open in Colab

- The policy is calculated from the value

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

where β is an inverse temperature

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

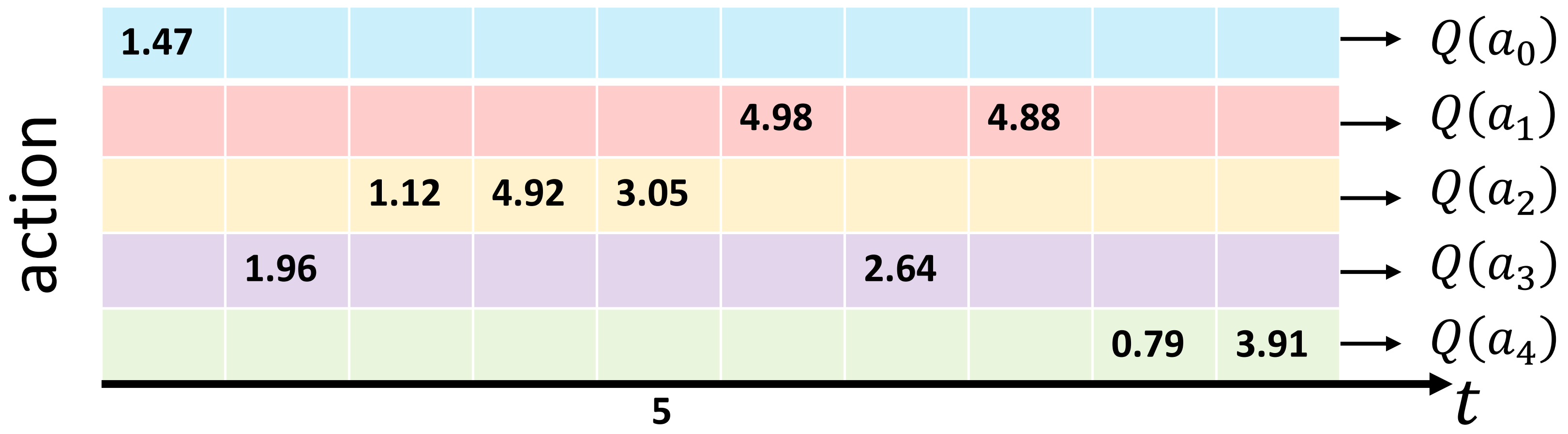


$$\beta = 0 \xrightarrow{\text{time}} \beta = \infty$$

(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



  - $R^1(a_2) = 1.12,\ Q^1(a_2) \triangleq R^1(a_2) = 1.12$
  - $R^2(a_2) = 4.92,\ Q^2(a_2) = \big(R^1(a_2) + R^2(a_2)\big)/2 = 3.02$
  - $R^3(a_2) = 3.05,\ Q^3(a_2) = \big(R^1(a_2) + R^2(a_2) + R^3(a_2)\big)/3 = 3.03$

Incremental Estimation of Q

$$Q^n(a) = \frac{1}{n}\sum_{i=1}^{n} R^i(a) = \frac{1}{n}\left(R^n(a) + \sum_{i=1}^{n-1} R^i(a)\right)$$

$$= \frac{1}{n}\left(R^n(a) + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R^i(a)\right)$$

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

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

Interpretation of the update rule

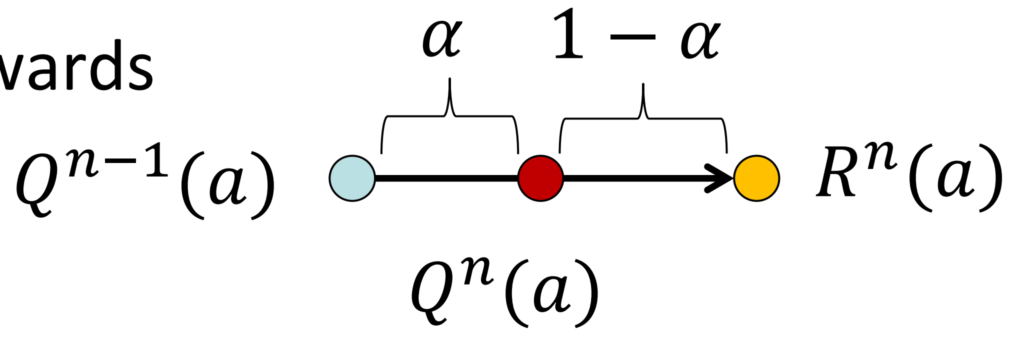
- $$\underbrace{Q^n(a)}_{\text{new estimate}} = Q^{n-1}(a) + \frac{1}{n}[\underbrace{R^n(a)}_{\text{target}} - \underbrace{Q^{n-1}(a)}_{\text{old estimate}}]$$

- $\delta \triangleq R^n(a) - Q^{n-1}(a)$ represents the error for the old estimate

- $\alpha \triangleq 1/n$ is a learning rate

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

- Online update rule for estimating rewards



A simple bandit algorithm

Initialize, for $a = 0$ to $k - 1$:

$$Q(a) \leftarrow 0, N(a) \leftarrow 0$$

Loop forever:

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

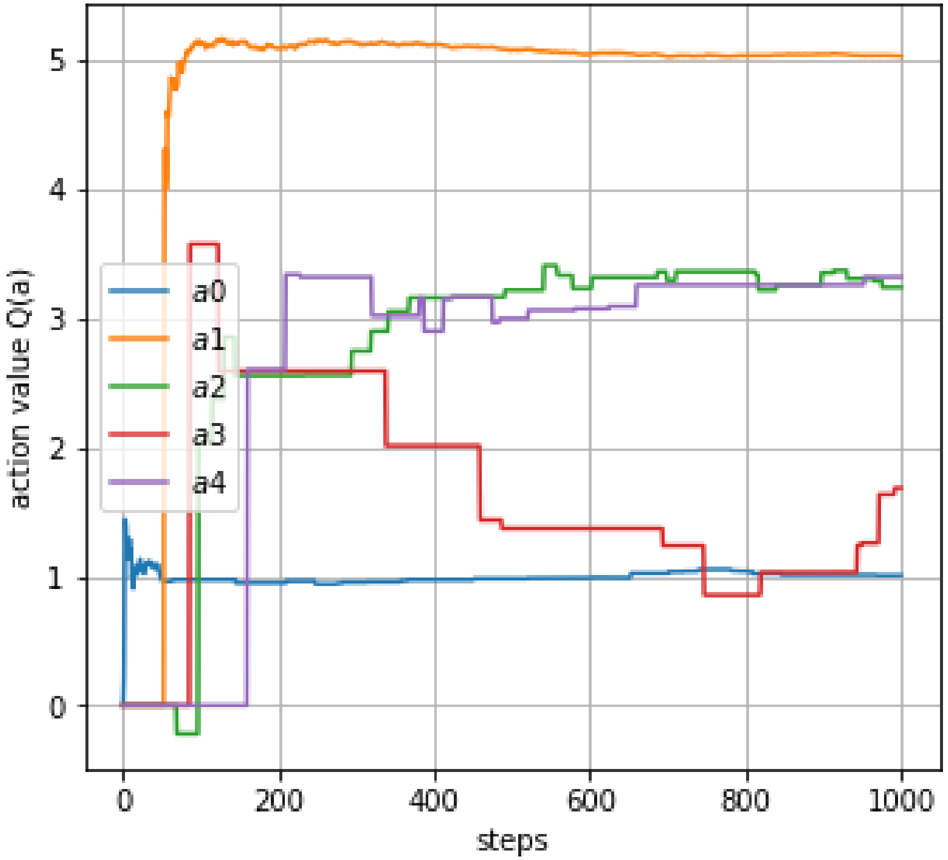
$$R \leftarrow \text{bandit}(A)$$

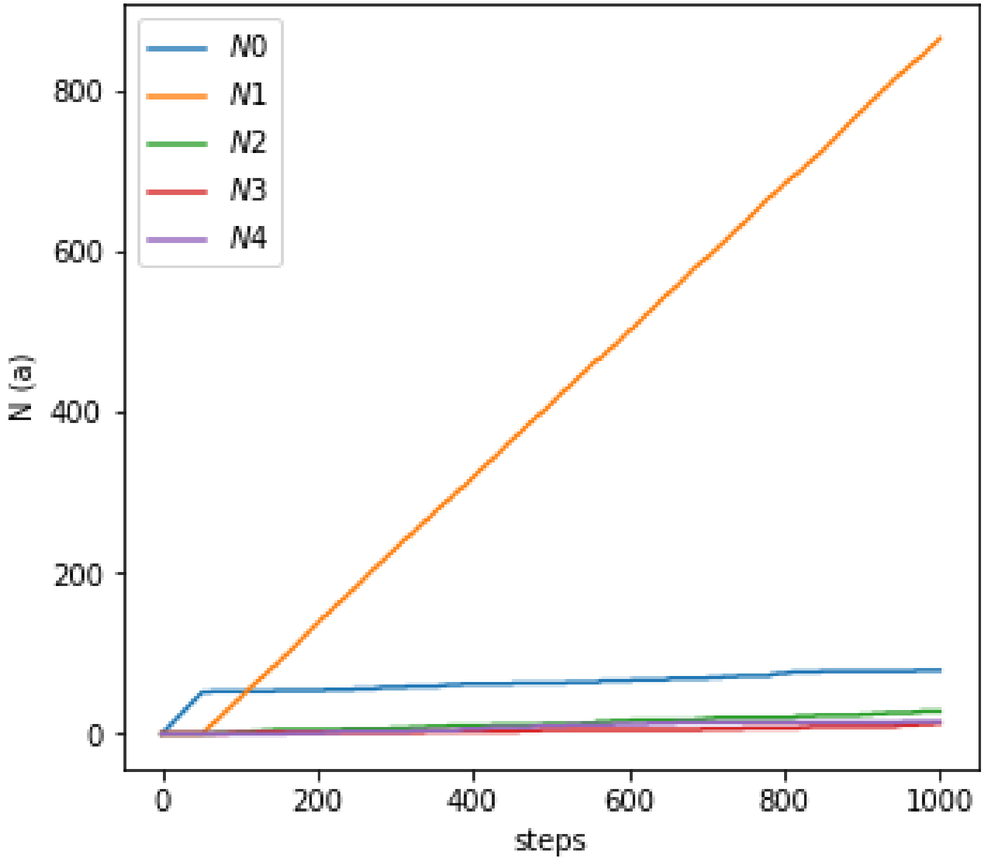
$$N(A) \leftarrow N(A) + 1 \qquad Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$$

Performance of ε-greedy policy (ε=0.1)

Open in Colab

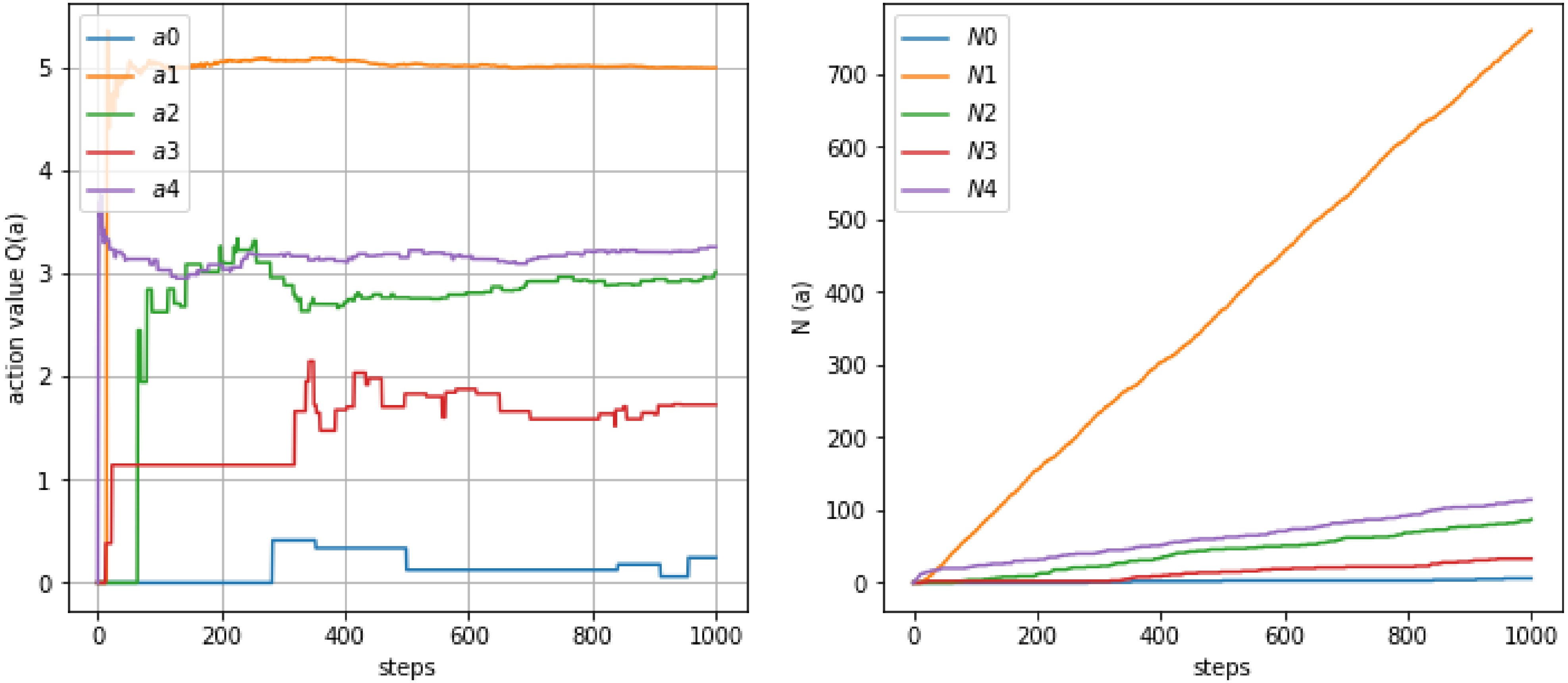
- Found the optimal action (a1) successfully





Performance of Boltzmann policy (β=1) Open in Colab

- Found the optimal action (a1) successfully



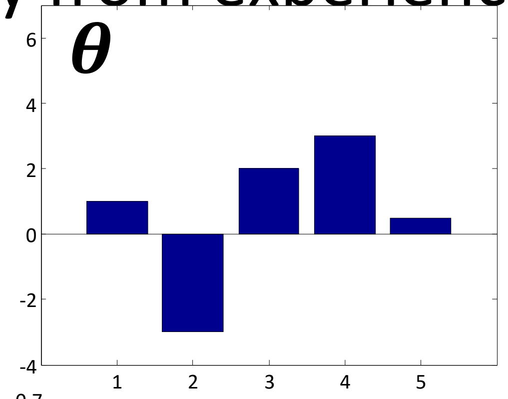
REINFORCEMENT LEARNING FOR BANDIT PROBLEMS
-- POLICY-BASED APPROACH --

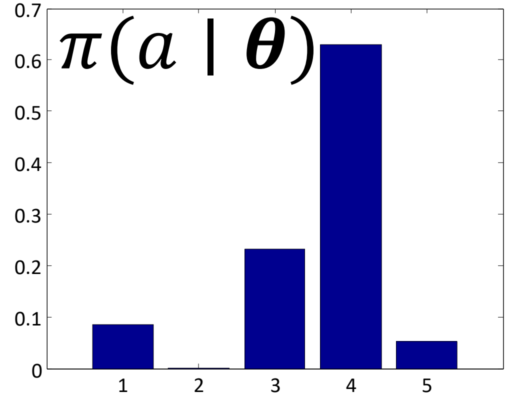
Policy-based approach

- 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

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

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

Policy-based approach

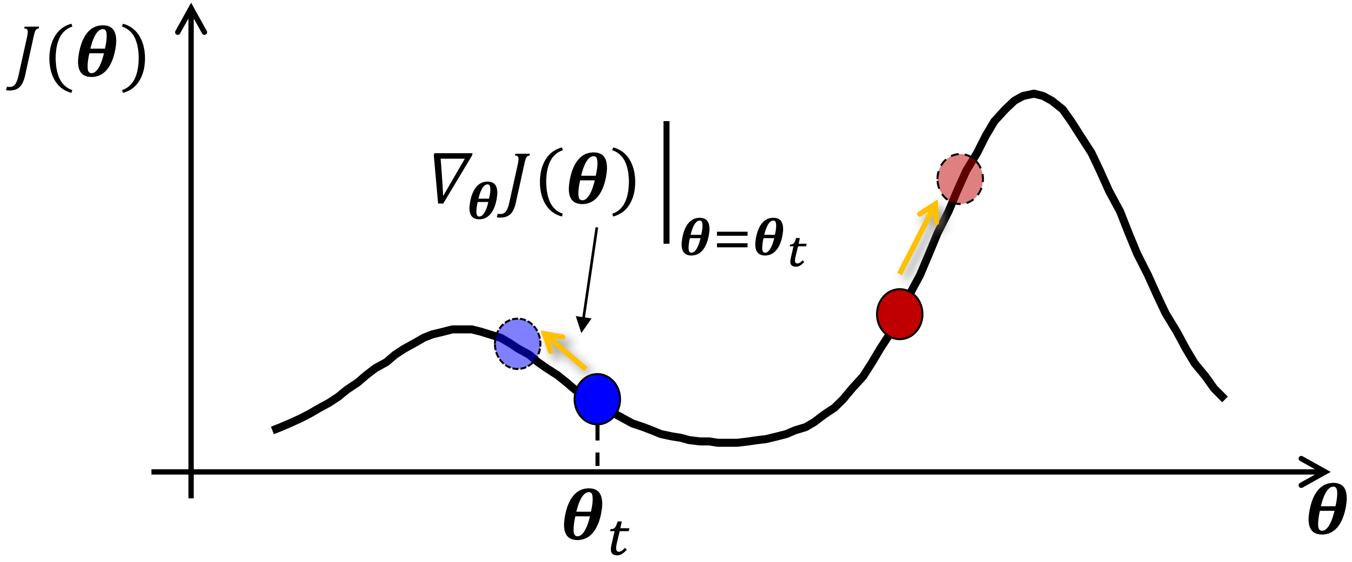
- The expected reward is

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[R] = \sum\nolimits_{r,a} r p(r \mid a) \pi(a \mid \boldsymbol{\theta})$$

- 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 α is a positive step-size parameter

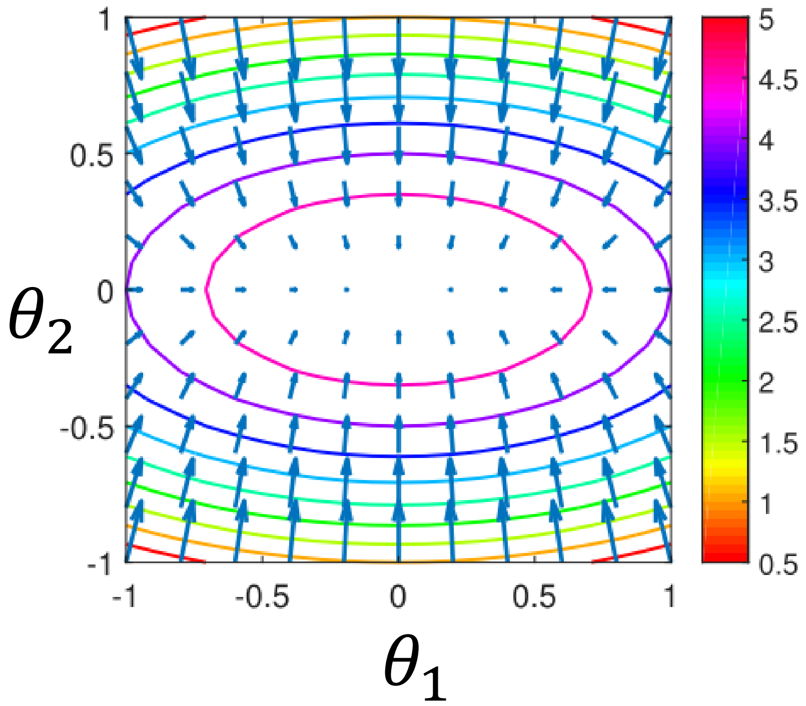


Gradient of function

Open in Colab

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

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



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

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

Gradient ascent optimization

- Find a local optimal solution

Initialize

$n \leftarrow 0$

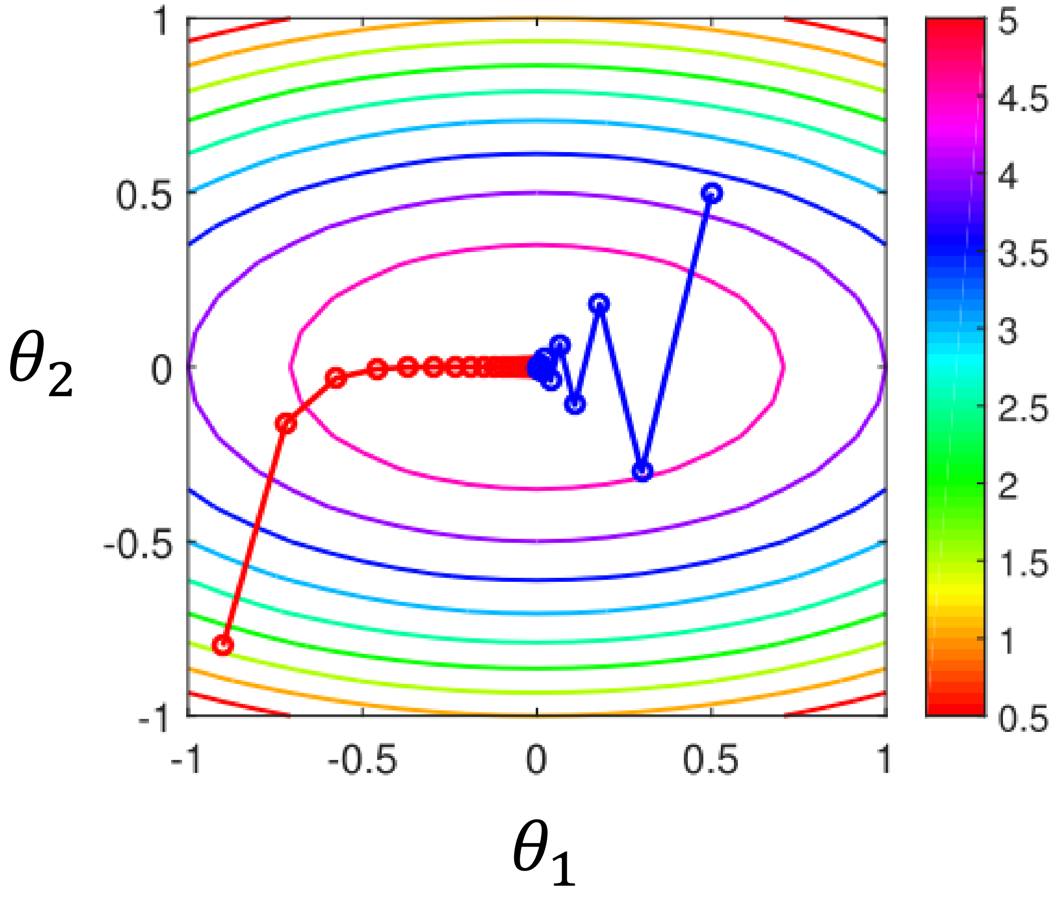
Repeat:

$\quad n \leftarrow n + 1$

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

until stopping_criterion($\boldsymbol{\theta}^n, \boldsymbol{\theta}^{n-1}, \epsilon$)

return $\boldsymbol{\theta}^n$



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

- $$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \left[\sum_{r,a} r p(r \mid a) \pi(a \mid \boldsymbol{\theta}) \right]$$
$$= \sum_{r,a} r p(r \mid a) \frac{\partial \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$
$$= \sum_{r,a} r p(r \mid a) \pi(a \mid \boldsymbol{\theta}) \frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$
$$= \mathbb{E}_{\pi} \left[R \frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right]$$

Calculate $\partial \ln \pi(\boldsymbol{a} \mid \boldsymbol{\theta}) / \partial \boldsymbol{\theta}$ (1/2)

- $\pi(a_i \mid \boldsymbol{\theta}) = \dfrac{\exp(\theta_i)}{\sum_{j=0}^{k-1} \exp(\theta_j)}$

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

- $$\frac{\partial \ln \pi(a_i \mid \boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{\pi(a_i \mid \boldsymbol{\theta})} \frac{\partial \pi(a_i \mid \boldsymbol{\theta})}{\partial \theta_i} = 1 - \pi(a_i \mid \boldsymbol{\theta})$$

Calculate $\partial \ln \pi(\boldsymbol{a} \mid \boldsymbol{\theta}) / \partial \boldsymbol{\theta}$ (2/2)

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

- $$\frac{\partial \ln \pi(a_i \mid \boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{\pi(a_i \mid \boldsymbol{\theta})}\frac{\partial \pi(a_i \mid \boldsymbol{\theta})}{\partial \theta_j} = -\pi(a_j \mid \boldsymbol{\theta})$$

Implementation

- Compute the gradient from experiences

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

- Use the stochastic gradient method

1. Sample action: $a \sim \pi(a \mid \boldsymbol{\theta})$

2. Receive the reward r from the environment: $r \sim p(r \mid a)$

3. Update θ by the stochastic gradient ascent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha r \frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

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

$$\mathbb{E}_\pi\left[(R-b)\frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]$$

$$= \underbrace{\mathbb{E}_\pi\left[R\frac{\partial \ln \pi(A \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]}_{J(\boldsymbol{\theta})} + b\underbrace{\mathbb{E}_\pi\left[\frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]}_{=0}$$

Proof sketch

- We want to prove $\mathbb{E}_{\pi}\left[\frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] = \mathbf{0}$

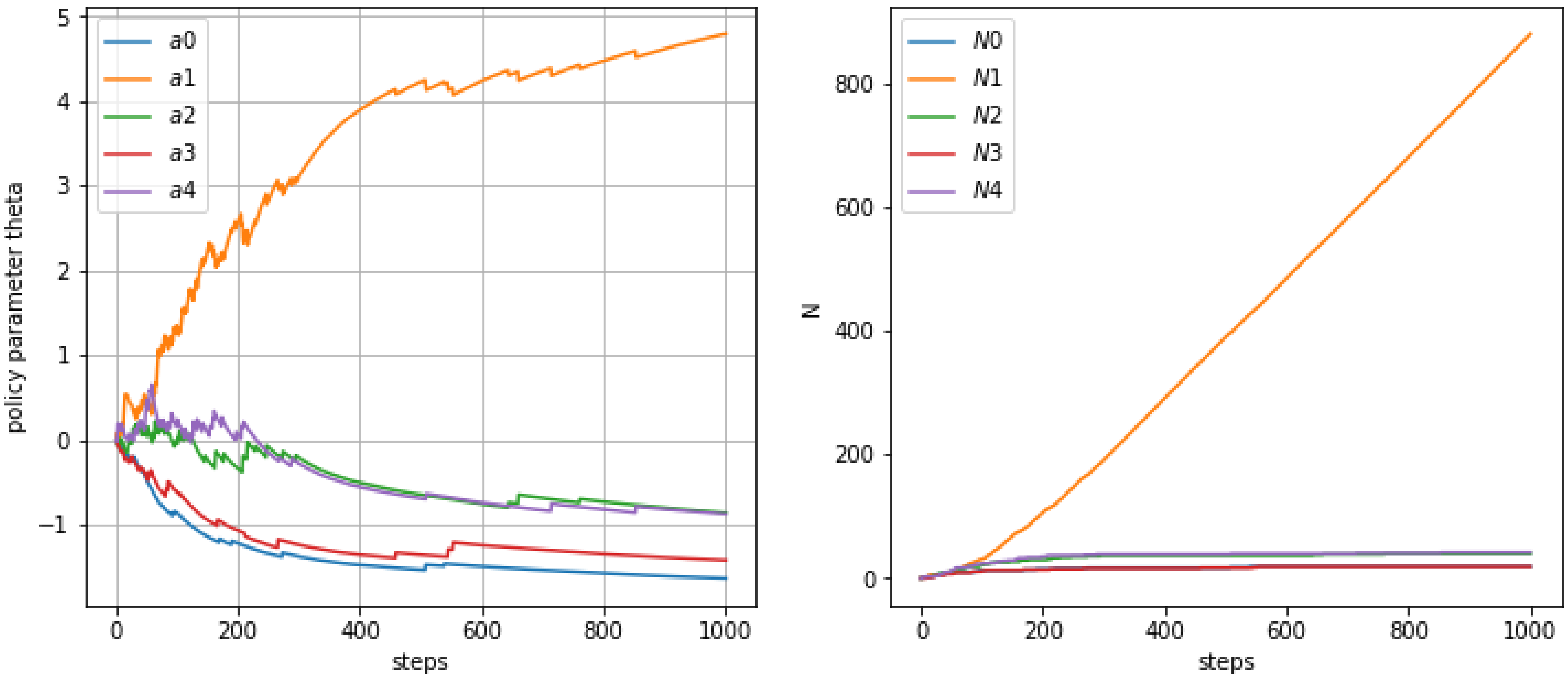
- $\mathbb{E}_{\pi}\left[\frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] = \sum_{r,a} p(r \mid a)\, \pi(a \mid \boldsymbol{\theta}) \frac{\partial \ln \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$

$$= \sum_{r,a} p(r \mid a) \frac{\partial \pi(a \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}}\Big[\underbrace{\sum_{r,a} p(r \mid a) \pi(a \mid \boldsymbol{\theta})}_{p(r,a \mid \boldsymbol{\theta})}\Big]$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}}[1] = \mathbf{0}$$

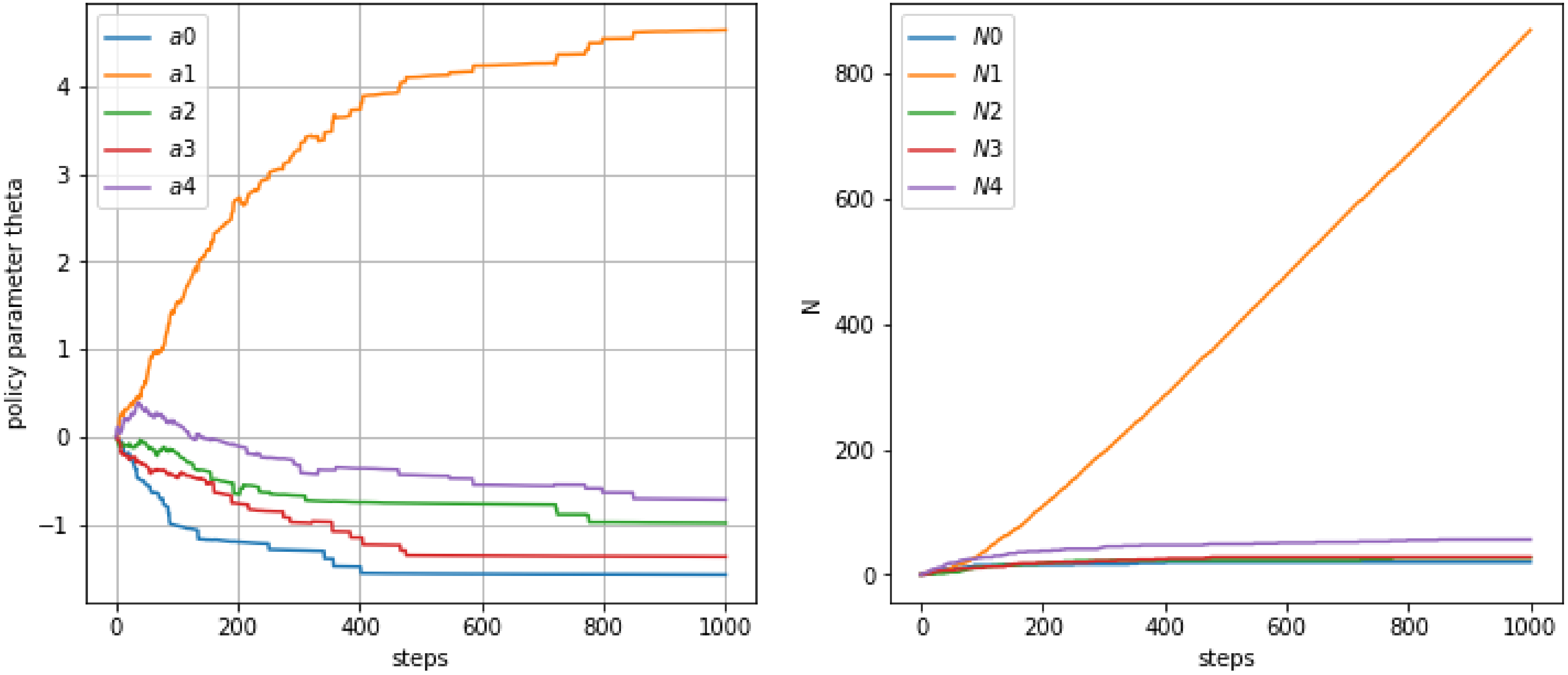
Performance without variance reduction

Open in Colab



Performance with variance reduction

Open in Colab



Summary

- Introduction to reinforcement learning
- Bandit problem
  - Action, Reward, Value
- Value-based approach: Learn the action-value function and derive the policy $\Rightarrow$ Indirect approach
- Policy-based approach: Learn the policy by the gradient ascent method of the objective function with respect to the policy parameters $\Rightarrow$ Direct approach