# Reinforcement Learning: From basics to Recent Algorithms

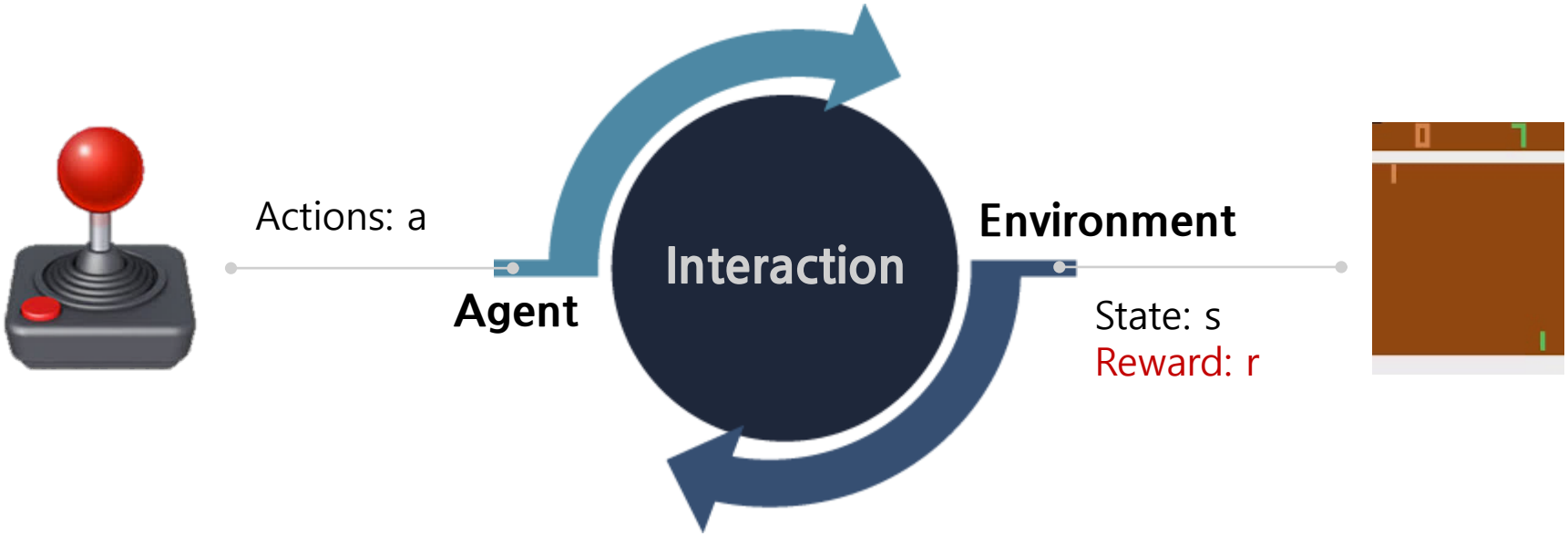
고려대학교
박영준

# Contents

- Reinforcement Learning
- Multi-Armed Bandits Problem
- Recent Algorithms
- Conclusions

# Reinforcement Learning (RL)

- Find policy $\pi(a|s)$ through $\max \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- It is called *approximate dynamic programming*



# Comparison with Supervised Learning

- Common
  - ✓ Predict something such as action or label
- Difference
  - ✓ Interaction

| | Reinforcement Learning | Supervised Learning |
|---|---|---|
| **Training Data** | $S, A, R, S, A, \ldots$ | $(X, Y)$ |
| **Model** | $\pi(a\|s)$ | $\hat{\mathrm{Y}} = F(\mathrm{X})$ |
| **Objective** | $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ | $\left(\mathrm{Y} - \hat{\mathrm{Y}}\right)^2$ |

# Training Data of Reinforcement Learning (Optional)

- Episode & S, A, R sequence

| Episode | Sequence |
|---|---|
| 1 | S, A, R, S, A, R, S, A, R, S, A, R |
| 2 | S, A, R |
| 3 | S, A, R, S, A, R |
| 4 | S, A, R, S, A, R, S, A, R |
| … | … |

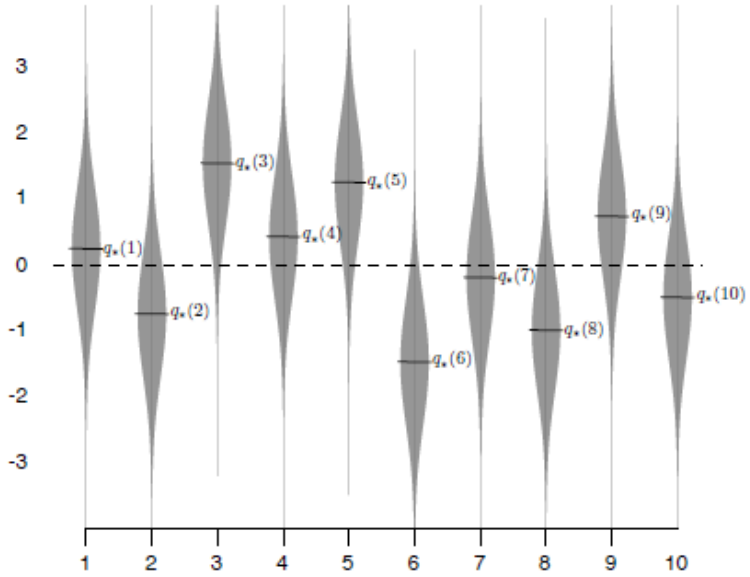
# Comparison with Unsupervised Learning

- Common
  - ✓ No labels
- Difference
  - ✓ RL maximize a cumulative rewards

| | Reinforcement Learning | Unsupervised Learning |
|---|---|---|
| **Training Data** | $S, A, R, S, A, \ldots$ | $X$ |
| **Model** | $\pi(a\|s)$ | $F(\mathrm{X})$ |
| **Objective** | $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ | $\|\mathrm{X} - F(X)\|$ |

# Multi-Armed Bandits Problem

- *k* slot machines
- State: single state [0, 0, 0, … 0]
- Action: choice a machine
- Reward: money



# Solutions for Multi-Armed Bandits

- **Action-value methods**

- **Gradient-based methods**

# Action-Value Function

- $Q_t(a)$ measure benefit of each action

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

| Bandit | Episode1 | Episode2 | Episode3 | Episode4 | Q |
|---|---|---|---|---|---|
| **1** | | | | 2 | 2 |
| **2** | | 3 | | | 3 |
| **3** | 5 | | | | 5 |
| **4** | | | 1 | | 1 |

# Action-Value Function

- $Q_t(a)$ measure preference of each action

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

| Bandit | Episode1 | Episode2 | Episode3 | Episode4 | Episode5 | Q |
|---|---|---|---|---|---|---|
| **1** | | | | 2 | 3 | 2.5 |
| **2** | | 3 | | | | 3 |
| **3** | 5 | | | | | 5 |
| **4** | | | 1 | | | 1 |

# Policy from Q Function

- Policy

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

| Bandit | Episode1 | Episode2 | Episode3 | Episode4 | Episode5 | Q |
|---|---|---|---|---|---|---|
| **1** | | | | 2 | 3 | 2.5 |
| **2** | | 3 | | | | 3 |
| **3** | 5 | | | | | 5 |
| **4** | | | 1 | | | 1 |

**Estimate Q function → Play (run policy)**

# Estimation of Action-Value

- Incremental update of $Q_t(a)$

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}
$$

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

*NewEstimate ← OldEstimate + StepSize (Target - OldEstimate)*

# Exploitation vs Exploration

- Adding $\epsilon$-greedy policy

**Action Trajectory**

| Bandit | Episode1 | Episode2 | Episode3 | Episode4 | Episode5 | Episode6 |
|---|---|---|---|---|---|---|
| **1** | | | | | 5 | |
| **2** | | 2 | | | | |
| **3** | 3 | | | | | |
| **4** | | | 1 | 1 | | |

**Q Trajectory**

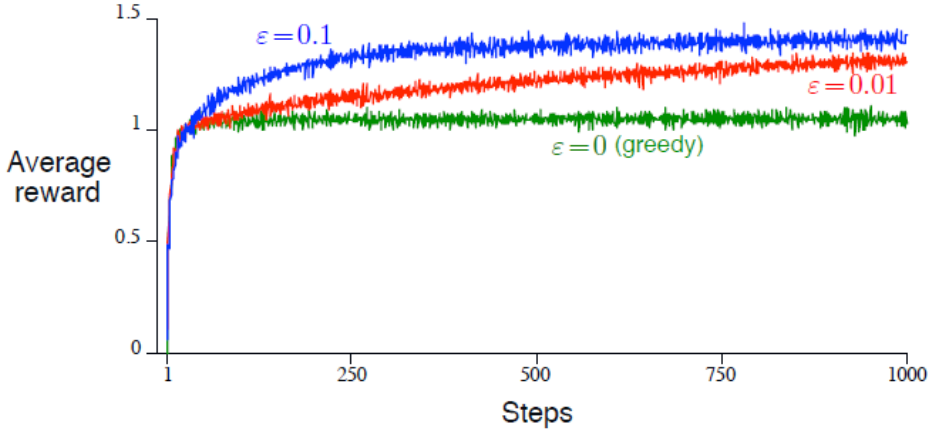
| Bandit | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | $Q_6$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 5 |
| 2 | 0 | 0 | 2 | 2 | 2 | 2 |
| 3 | 0 | 3 | **3** | 3 | 3 | 3 |
| 4 | 0 | 0 | 0 | 1 | 1 | 1 |

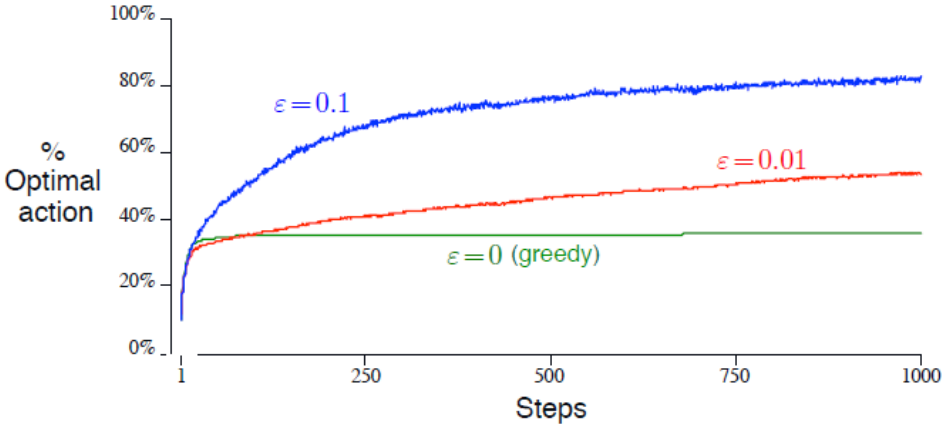
# Effect of $\epsilon$-greedy Policy

- $\epsilon$-greedy policy

Initialize, for $a = 1$ to $k$:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$



# Gradient-Based Methods

- Softmax policy (also called as Gibbs or Boltzmann)

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Preference for each action $H_t(a)$

- Update $H_t(a)$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t,$$

  - If the reward is higher than the baseline, then the probability of taking the action in the future is increased
  - if the reward is below baseline, then probability is decreased.

# Update Rule of Gradient-Based Methods

- Gradient ascent algorithm

$$H_{t+1}(a) \leftarrow H_t(a) + \alpha \frac{\partial E[R_t]}{\partial H_t(a)}$$ Objectives

$$\frac{\partial E[R_t]}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \left[ \sum_x \pi_t(x)\, q_*(x) \right] \qquad E[R_t] = \sum_x \pi_t(x)\, q_*(x)$$

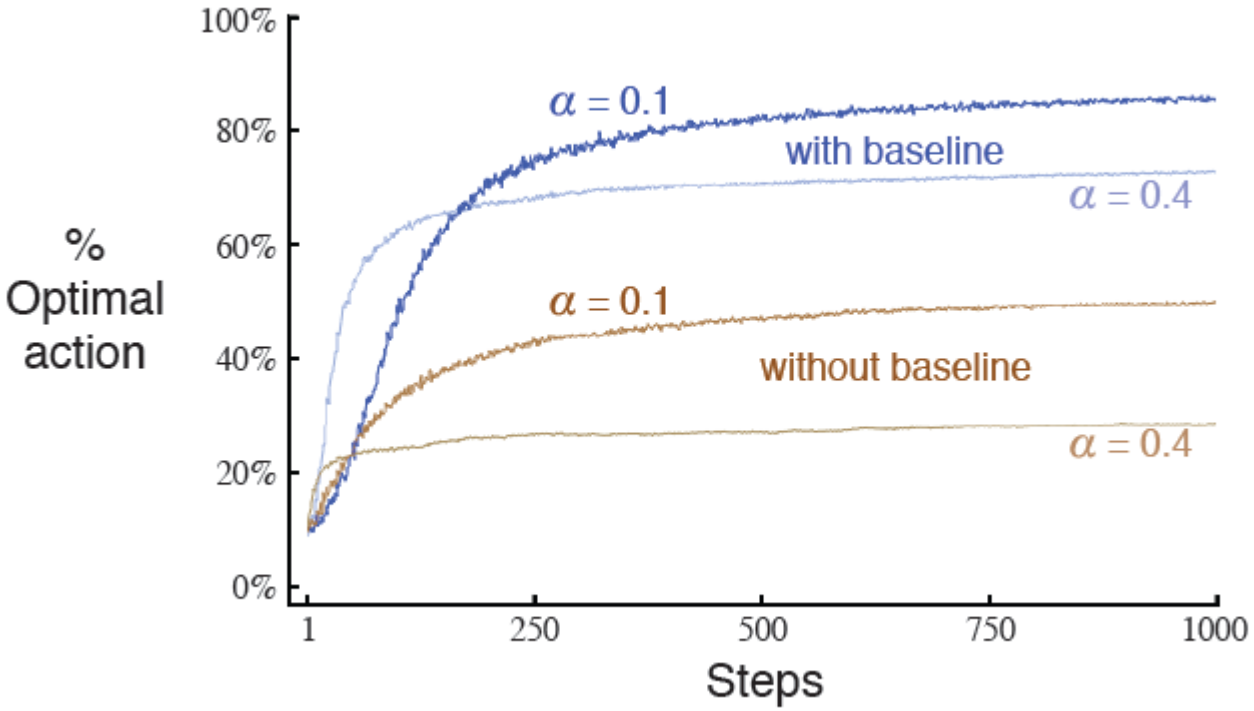
$$= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

$$= \sum_x R_t \frac{\partial \pi_t(x)}{\partial H_t(a)} \qquad E[R_t|A_t] = q_*(A_t)$$

- Details in page 38~40

# Gradient-Based Methods

- Performance



# Connection Between Reinforcement Learning

- **Action-value methods**

  → **Value-based RL**

  ✓ DQN

- **Gradient-based methods**

  → **Policy-based RL**

  ✓ Policy Gradient

- **Action-value methods & Gradient-based methods**

  → **Actor-Critic methods** (actor: policy-based / critic: value-based)

  ✓ A2C, A3C,…

# Beyond Multi-Armed Bandits

- Long episodes
- Large states & actions

Breakout and Space Invaders, 2 of the 49 Atari games used in the paper


# Value Functions

- State-value functions

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

- Action-value functions

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

# Bellman Equations

- Bellman Eq. for $v_\pi$

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \Big[ r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \,|\, s, a) \Big[ r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

- Bellman **Optimality** Eq.

$$
\begin{aligned}
v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s', r} p(s', r \,|\, s, a) \Big[ r + \gamma v_*(s') \Big]
\end{aligned}
$$

$$
\begin{aligned}
q_*(s, a) &= \mathbb{E}\Big[ R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a \Big] \\
&= \sum_{s', r} p(s', r \,|\, s, a) \Big[ r + \gamma \max_{a'} q_*(s', a') \Big],
\end{aligned}
$$

# Dynamic Programming

- Policy iteration, value iteration

**Value Iteration, for estimating $\pi \approx \pi_*$**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$

# Model-Free Methods

- Monte Carlo (MC) Methods
  - S, A, R, S, A, R, S, A, R, S, A, R, Terminate → *Update*
  - S, A, R, S, A, R, Terminate → *Update*

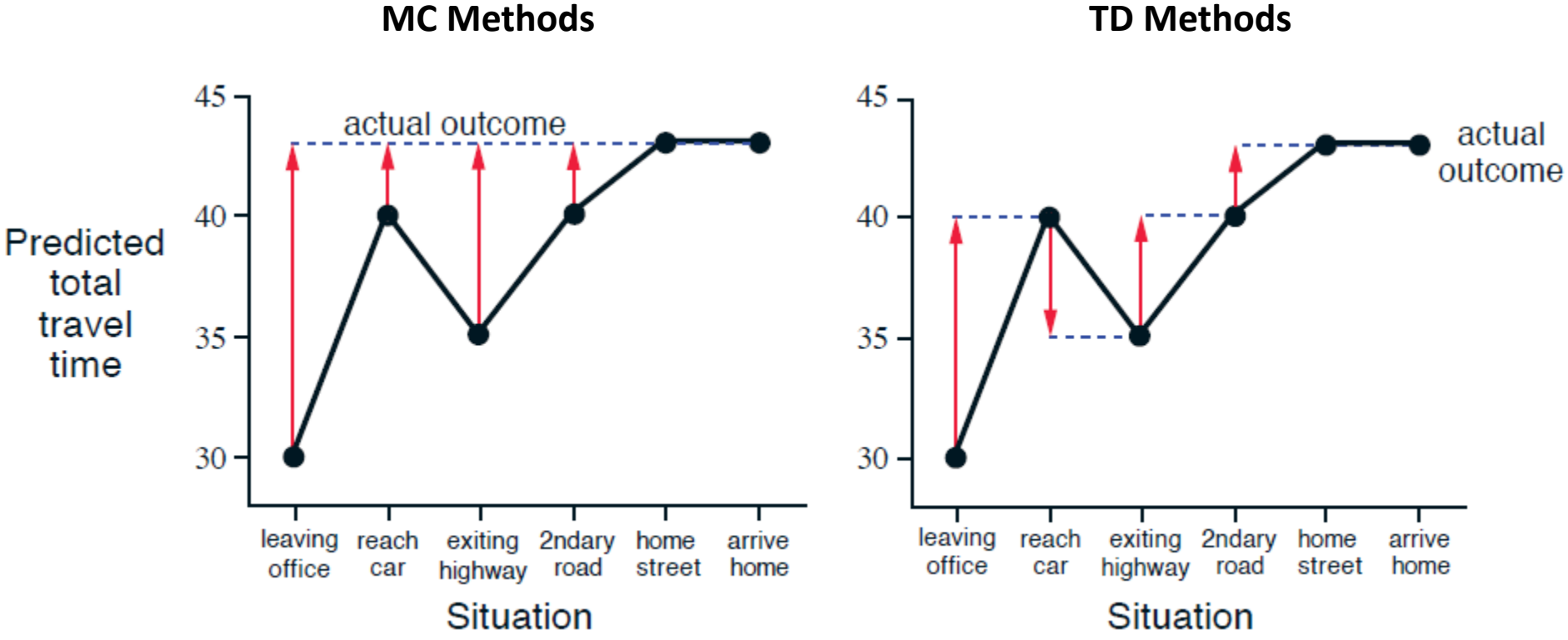
$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

- Temporal Difference (TD) Learning
  - S, A, R, *Update*, S, A, R, *Update*, S, A, R, *Update*, S, A, R, Terminate, *Update*
  - S, A, R, *Update*, S, A, R, Terminate, *Update*

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

*NewEstimate ← OldEstimate + StepSize (Target - OldEstimate)*

# MC vs TD


MC Methods
TD Methods
Predicted total travel time
45
40
35
30
actual outcome
actual outcome
leaving office
reach car
exiting highway
2ndary road
home street
arrive home
Situation
Situation

# On-Policy vs Off-policy

- To enhance exploration, use off-policy methods
  - One policy (target policy) learn (*exploitation*)
  - Another policy generate behavior (*exploration*)

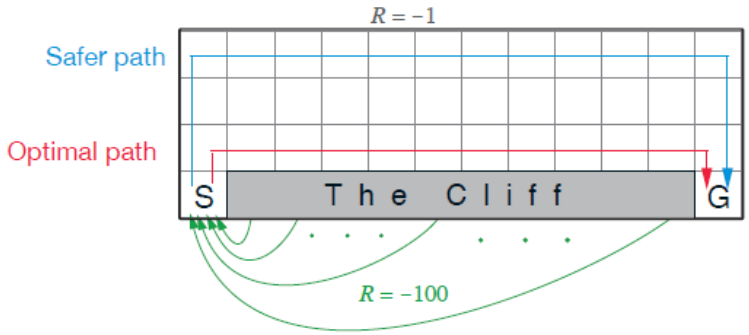
- SARSA (on-policy TD)

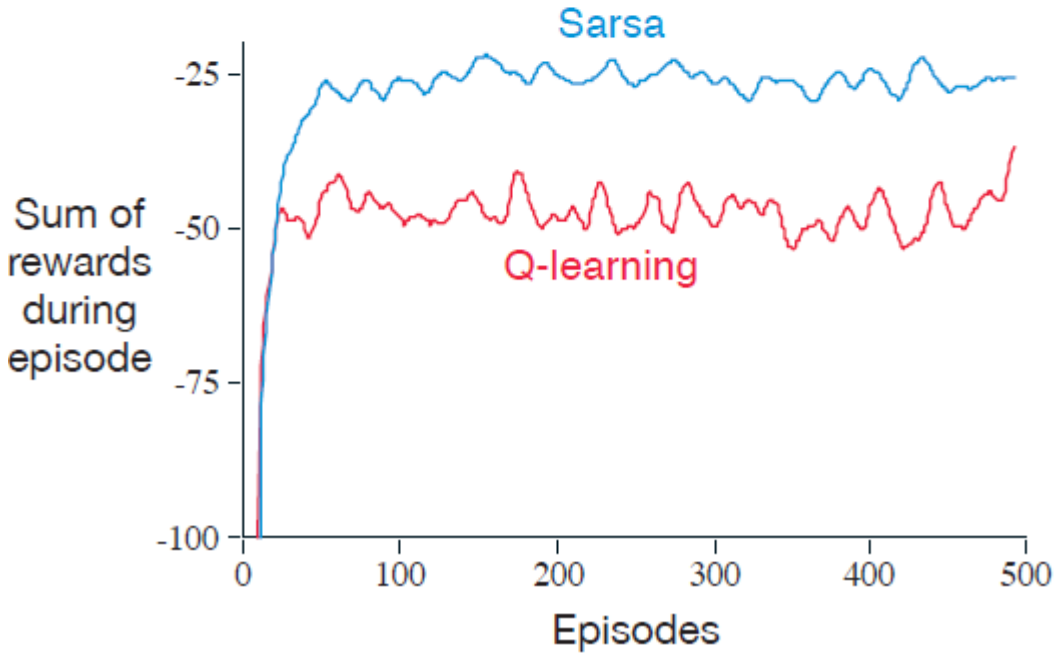
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

- Q-learning (off-policy TD)

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

# On-Policy vs Off-policy



# Deep Reinforcement Learning

- Model-free control

## Deep Q-Learning (DQN)

$$q(s,a) = E[G_t|S_t = s, A_t = a]$$
$$\approx F(s,a|\theta)$$

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\, q(S_{t+1}, a')$$

- Return의 기대값 Q value를 deep neural network ($F$)를 이용해 근사
- 매 state마다 가장 큰 Q value에 대응하는 action을 선택

## Policy Gradient

$$\pi(S_{t+1}) \approx F(s|\theta)$$

$$\nabla J(\theta) = \nabla E[G_t|\pi]$$

- Policy를 deep neural network ($F$)를 이용해 근사
- Policy가 주어졌을 때, return의 기대값이 목적함수

# DQN

- Off-policy TD learning

## Double DQN

- **Objective**

$$\mathcal{L}(\theta) = [y^{target} - Q(s_t, a_t; \theta)]^2$$

- **DQN**

$$y^{target} = r + \gamma \max_a Q(s_{t+1}, a; \theta^-)$$

- **Double DQN**

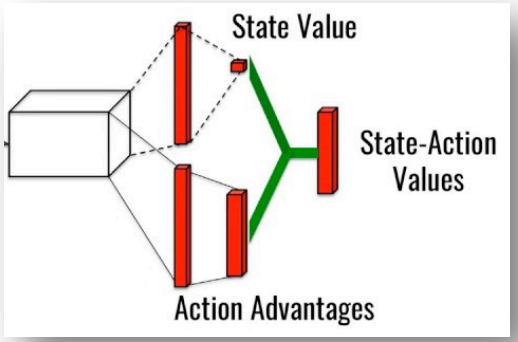
$$y^{target} = r + \gamma Q(s_{t+1}, \operatorname{argmax}_a Q(s_{t+1}, a; \theta); \theta^-)$$

## Dueling DQN

- **Q value = state value + advantage**

$$Q(s, a; \theta, \alpha, \beta)$$

$$= V(s; \theta, \beta) + \left[A(s, a; \theta, \alpha) - \max_{a'} A(s, a'; \theta, \alpha)\right]$$



# Policy Gradient Methods

- REINFORCE (*MC policy gradient*)

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(s)\, G_t$$

- Actor Critic Policy Gradient (*mix*)

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(s)\, Q(s,a)$$

- Advantage Actor-Critic Policy Gradient (*mix*)

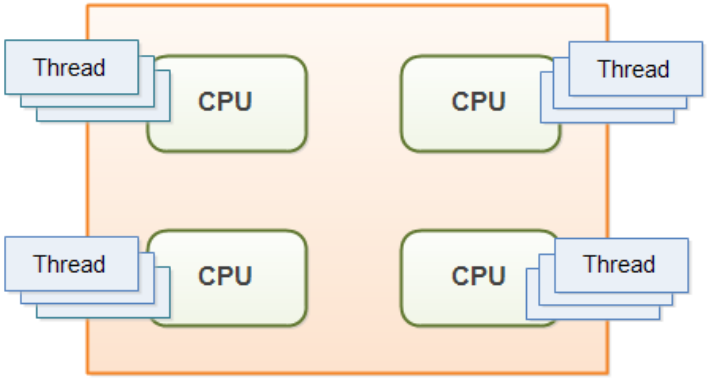
$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(s)\, [Q(s,a) - V(s)]$$

# Tricks on Deep RL

- Value-based method uses **memory**
  - S, A, R, *Update*, S, A, R, *Update*, S, A, R, *Update*, S, A, R, Terminate, *Update*

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

- **Asynchronous** methods



# Conclusions

[ 고전의 지혜 – 논어 명언명구 ]

옛것을 익혀서 새것을 알다

온고지신(溫故知新)