# Author

(ICML) 2017

## A Distributional Perspective on Reinforcement Learning

**Marc G. Bellemare** [*1] **Will Dabney** [*1] **Rémi Munos** [1]

Marc G. Bellemare

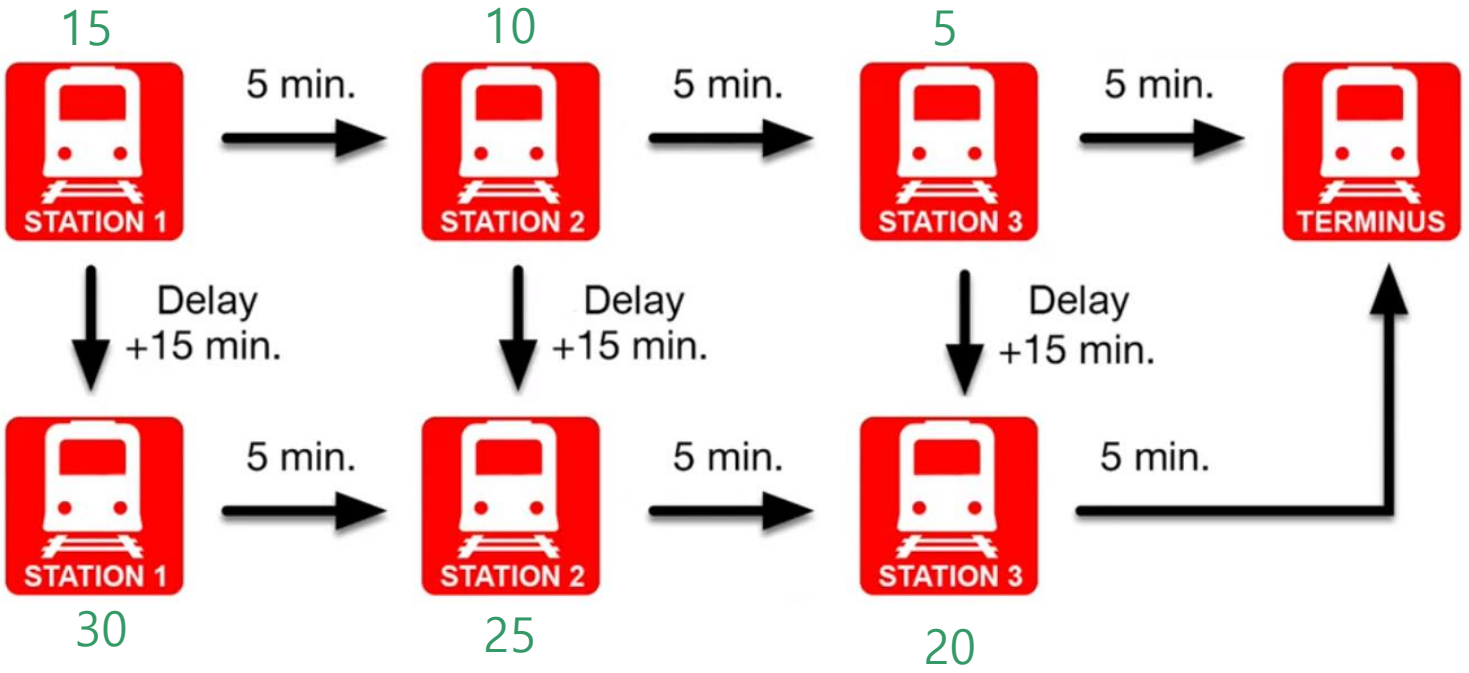
팔로우

Google Brain
google.com의 이메일 확인됨 - 홈페이지

Reinforcement Learning  Information Theory

| 제목 | 인용 | 연도 |
|---|---|---|
| Human-level control through deep reinforcement learning<br>V Mnih, K Kavukcuoglu, D Silver, AA Rusu, J Veness, MG Bellemare, ...<br>nature 518 (7540), 529-533 | 12534 | 2015 |
| The Arcade Learning Environment: An Evaluation Platform for General Agents<br>MG Bellemare, Y Naddaf, J Veness, M Bowling<br>Journal of Artificial Intelligence Research 47, 253--279 | 1641 | 2013 |
| Unifying count-based exploration and intrinsic motivation<br>M Bellemare, S Srinivasan, G Ostrovski, T Schaul, D Saxton, R Munos<br>Advances in neural information processing systems 29, 1471-1479 | 664 | 2016 |
| A distributional perspective on reinforcement learning<br>MG Bellemare, W Dabney, R Munos<br>arXiv preprint arXiv:1707.06887 | 510 | 2017 |

# Motivation




**Breaks down once a week (5일에 한번 delay)**

**(3x5) + 15 / 5 = 18 min.**

미래 보상들은 complex, Multimodal의 특성을 가지는데,
기대값은 각 보상들이 가지는 특성을 담아내지 못한다.
→ 평균 하지말고 case마다의 시간 계산하자

# DQN

Part 02

## BackGround

[논문 요약]

- ✓ 시간 t에서 Discount Factor($\gamma$)가 적용된 보상 $r_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (T는 게임의 종료 시점)
  ex) $r_3 = r_3 + \gamma^1 r_4 + \gamma^2 r_5 + \gamma^3 r_6 + \ldots + \gamma^{T-3} r_T$
- ✓ 새로 정의한 action-value function $Q^*(s,a) = max_\pi E[R_t | s_t = s, a_t = a, \pi]$ ($s_t$는 Sequence, a는 선택한 Action)
  즉, Policy $\pi$를 통해 얻을 수 있는 Reward 의 Maximum Expected Value를 구함
- ✓ Bellman Equation에 따라 all possible action a'에 대해 next time-step에서 Optimal $Q^*(s,a)$를 안다면
  Optimal Strategy는 $r + \gamma * Q^*(s,a)$를 Maximize하는 것이다.
- ✓ RL은 위와 같은 Optimal Strategy를 iterative update하게 사용, $\text{converge}(Q_i \rightarrow Q^*) \text{ as } (i \rightarrow \infty)$

$$Q_{i+1}(s,a) = \mathbb{E}\left[ r + \gamma Q_i(s',a') \mid s,a \right]$$

그러나 action-value function은 각 sequence마다 독립적으로 측정되기 때문에, 이런 방식은 실제로 impractical(비현실적)하다. 대신 function approximator를 사용하여 action-value function을 적절히 approximate 시킨다.

$$Q(s,a;\theta) \simeq Q^*(s,a).$$

일반적으로는 Linear Function으로 Approximate 하지만, 간혹 Non-Linaer Function으로 Approximate하는 경우도 있다.
neural network function approximator로 weight $\theta$ 를 사용하는 것을 Q-network라고 한다. Q-Network는 each iteration마다 바뀌는 Loss function $L_i(\theta_i)$를 최소화시킴으로써 학습을 한다.

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[ \left( y_i - Q(s,a;\theta_i) \right)^2 \right], \text{where, } y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[ r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) \mid s,a \right]$$

- ✓ But 이러한 Action-Value Function은 각 Sequence에 대해 독립적으로 estimate되므로 비현실적이다.
- ✓ $\theta$를 Weight로 갖는 Non-Linear Network Function Approximator를 통해 Approximate $Q(s,a;\theta) \simeq Q^*(s,a)$.
- ✓ $\theta$를 학습시키기 위해 i번째 iteration에서 다음과 같은 loss function을 갖게 하여
  $\theta$가 Converge 할 때 까지 반복시키고, 여기서 $y_i$는 iteration의 target value이다.

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[ \left( r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i) \right) \nabla_{\theta_i} Q(s,a;\theta_i) \right]$$

- ✓ $\theta_{i-1}$ 은 $L_i(\theta_i)$ 를 optimizing 할 때 정해진다.

# Key Idea

Return을 Distribution으로 만들어 Randomness한 특성과 정보를 최대한 반영해보자

**Bellman Equation**
(State Action Function)

$$Q(x,a) = \mathbb{E}\,R(x,a) + \gamma\,\mathbb{E}\,Q(X',A').$$

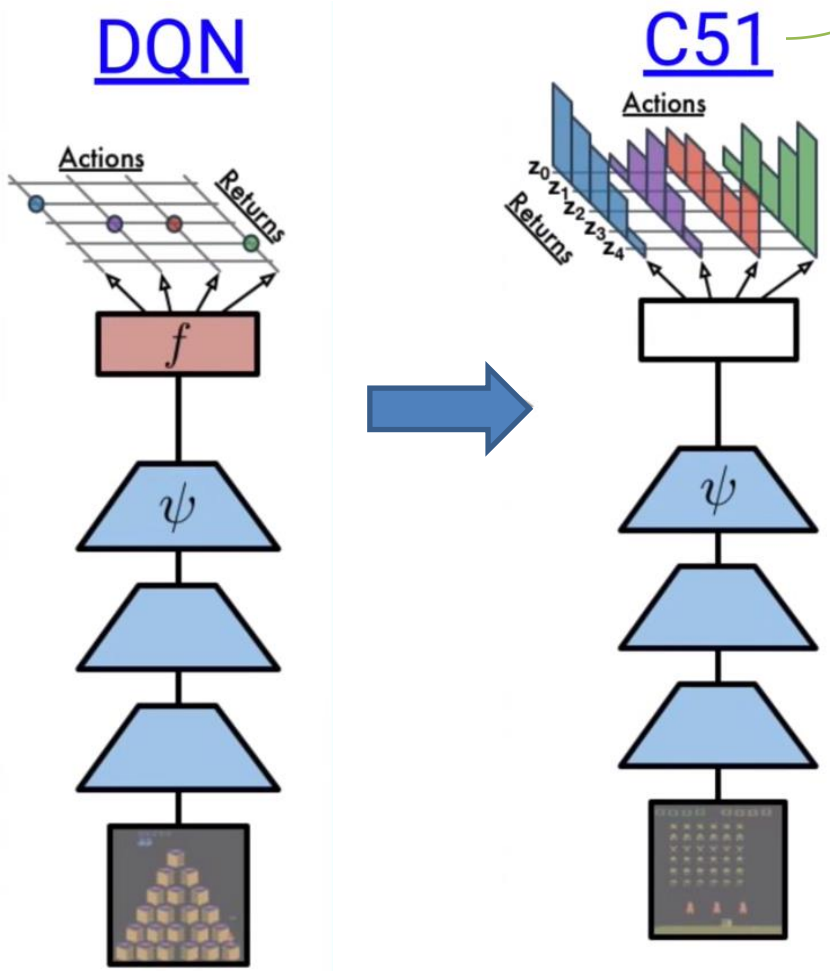
**distributional bellman equation**

$$Z(x,a) \overset{D}{=} R(x,a) + \gamma Z(X',A').$$

$$Q(x,a) = \cancel{\mathbb{E}}\,R(x,a) + \gamma\cancel{\mathbb{E}}\,Q(X',A').$$

기존에는 Q의 기대값을 러닝했다면, Random Return Z의 Distribution을 러닝

# C51



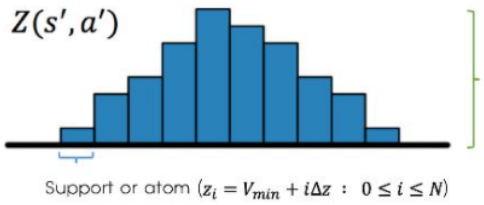

Action마다
Discrete Distribution with 51 values

**4.1. Parametric Distribution**

We will model the value distribution using a discrete distribution parametrized by $N \in \mathbb{N}$ and $V_{\text{MIN}}, V_{\text{MAX}} \in \mathbb{R}$, and whose support is the set of atoms $\{z_i = V_{\text{MIN}} + i\triangle z : 0 \leq i < N\}$, $\triangle z := \frac{V_{\text{MAX}} - V_{\text{MIN}}}{N-1}$. In a sense, these atoms are the "canonical returns" of our distribution. The atom probabilities are given by a parametric model $\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^N$

$$Z_\theta(x, a) = z_i \quad \text{w.p.} \quad p_i(x, a) := \frac{e^{\theta_i(x,a)}}{\sum_j e^{\theta_j(x,a)}}.$$

The discrete distribution has the advantages of being highly expressive and computationally friendly (see e.g. Van den Oord et al., 2016).



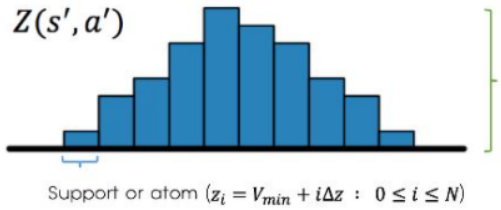

$N$: support 의 수

$V_{min}, V_{max}$: support 의 최소, 최대값

$p_i$ : 각 support의 값을 받을 확률 $(0 \leq i \leq N)$

-> Network 의 output 을 softmax

# Target Value Distribution




$N$: support 의 수

$V_{min}, V_{max}$: support 의 최소, 최대값

$p_i$ : 각 support의 값을 받을 확률 $(0 \le i \le N)$

-> Network 의 output 을 softmax

$$Q(x_t, a_t) = \sum_i z_i p_i(x_t, a_t)$$

$N$: supports의 수, $V_{min}$: support의 최소값, $\Delta z$: support간 간격

$z_i$: support $(z_i = V_{min} + i\Delta z : \ 0 \le i \le N)$

$p_i$: support $z_i$에 대한 확률

- Target value distribution

1. 다음 state에 대한 value 분포를 구함 $Z(s', a') \longrightarrow p_i(s', a')$
2. Target atom을 구함 $\hat{T} z_i$

$$\hat{T} z_i = [R + \gamma z_i]_{V_{\min}}^{V_{\max}}$$

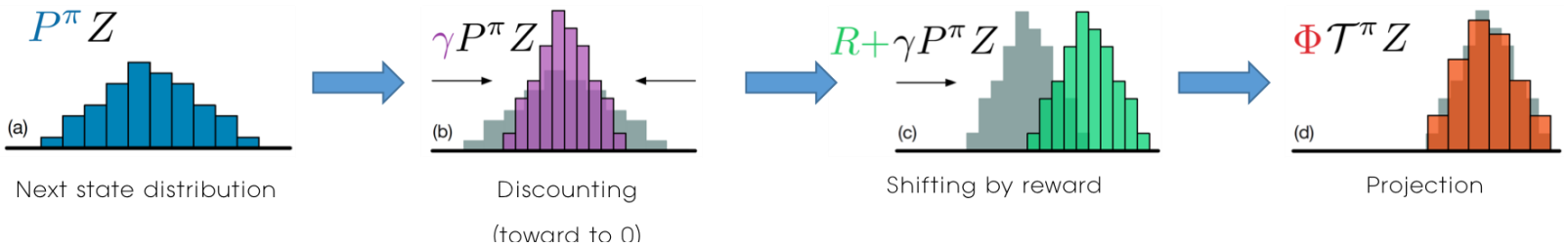

Next state distribution

Discounting (toward to 0)

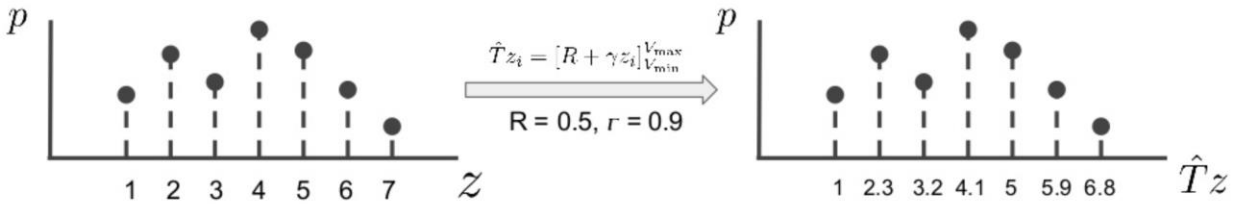
Shifting by reward

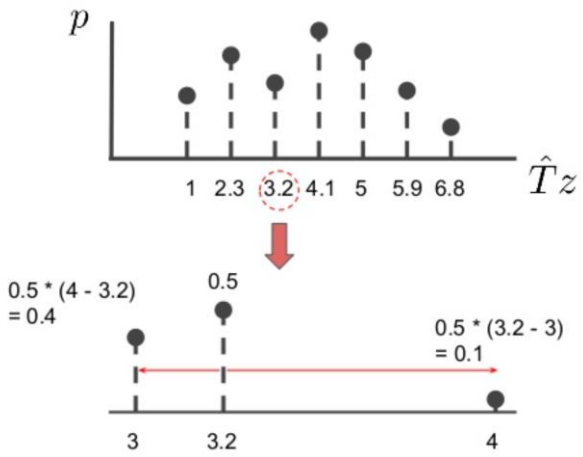
Projection

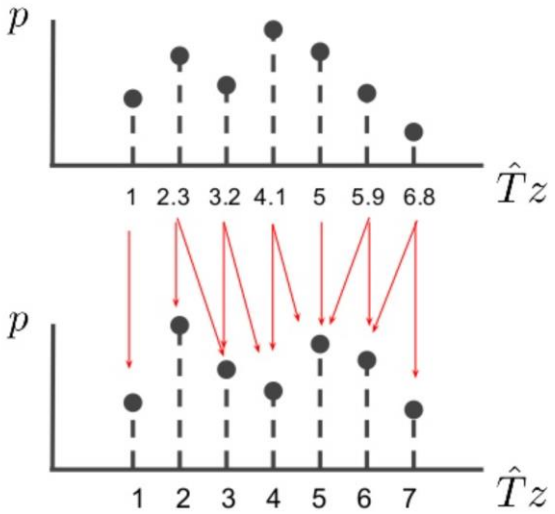
# Target Value Distribution - Projection




- Target value distribution - Projection
  - Target value distribution과 value distribution의 atom이 불일치
    - *Target atom이 reward와 r와의 연산 때문에 변경됨*
  - **Projection**을 통해 일치 시킴.

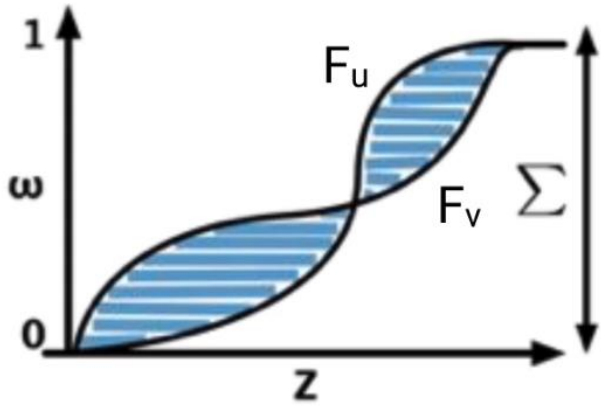


# Cost function

Target value distribution과 현재 value distribution간의 차이를 줄이는 방향으로 학습




Wasserstein

$$d_p(F,G) = \left( \int_0^1 \left| F^{-1}(u) - G^{-1}(u) \right|^p du \right)^{1/p}$$

# Algorithm

Replay Buffer에서
Batch size만큼 추출

Projection Distribution
(분포 만들기)

KL-divergence(cross entropy)
로 Loss 구하기

**Algorithm 1** Categorical Algorithm

**input** A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0,1]$
  $Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$
  $a^* \leftarrow \arg\max_a Q(x_{t+1}, a)$
  $m_i = 0, \quad i \in 0, \ldots, N-1$
  **for** $j \in 0, \ldots, N-1$ **do**
    # Compute the projection of $\hat{\mathcal{T}} z_j$ onto the support $\{z_i\}$
    $\hat{\mathcal{T}} z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$
    $b_j \leftarrow (\hat{\mathcal{T}} z_j - V_{\text{MIN}})/\Delta z$ # $b_j \in [0, N-1]$
    $l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$ ⟶ $l$: 버림, $u$: 올림
    # Distribute probability of $\hat{\mathcal{T}} z_j$
    $m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$
    $m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$
  **end for**
**output** $-\sum_i m_i \log p_i(x_t, a_t)$ # Cross-entropy loss

- Distribution의 기대값을 통해 Q fuction 을 계산
- Q값을 최대로 하는 action 선택

- Target distribution 계산
- Target distribution 각각에 해당하는 support 결정

- Target distribution 을 기존 support 에 분배

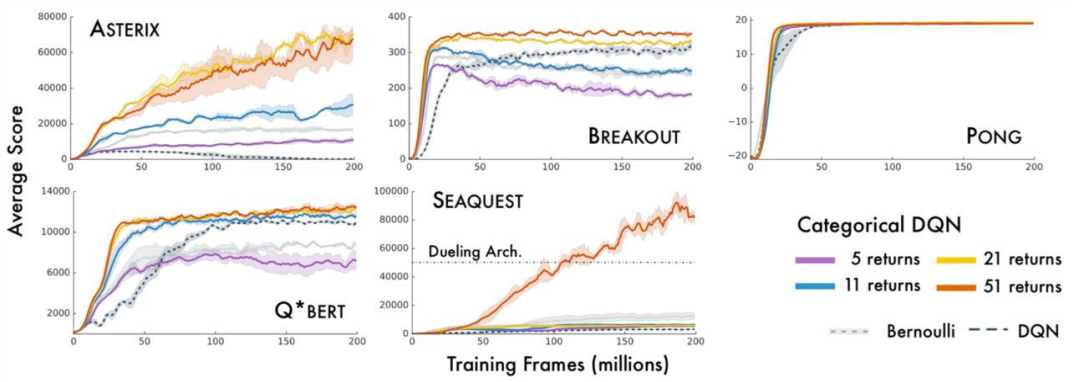
- Target distribution 과 예측된 distribution 간 cross entropy loss

# Result

**Categorical DQN**

Support의 수(N)을 많이 설정할수록 더 좋은 성능을 보인다.

•51개의 support를 사용했을 때 게임에서 좋은성능을 보여 C51이라 지칭.




•Sparse한 reward환경의 게임 (Private Eye, Venture)같은 게임에서 다른 알고리즘에 비해 좋은성능을 보임

•기존 DQN, DDQN, Nueling, PER, PER + Dueling 과 비교 했을 때 매우 좋은 성능을 보여준다.

| GAMES | RANDOM | HUMAN | DQN | DDQN | DUEL | PRIOR. DUEL. | C51 |
|---|---|---|---|---|---|---|---|
| Private Eye | 24.9 | **69,571.3** | 146.7 | 129.7 | 103.0 | 206.0 | 15,095 |
| Venture | 0.0 | 1,187.5 | 163.0 | 98.0 | 497.0 | 48.0 | **1,520** |

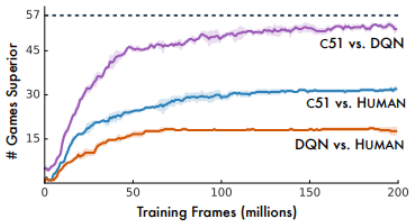



*Figure 12.* Number of Atari games where an agent's training performance is greater than a baseline (fully trained DQN & human). Error bands give standard deviations, and averages are over number of games.

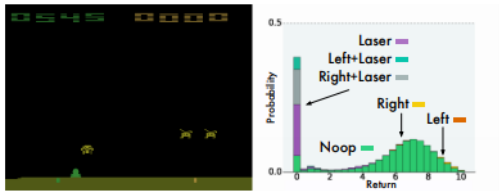



*Figure 4.* Learned value distribution during an episode of SPACE INVADERS. Different actions are shaded different colours. Returns below 0 (which do not occur in SPACE INVADERS) are not shown here as the agent assigns virtually no probability to them.

# Reference

https://www.youtube.com/watch?v=6CdV0kgyPI8&t=603s
https://www.youtube.com/watch?v=fwv8iEifCFU
https://kminseo.tistory.com/16