Reinforcement Learning - 2

AI & OPTIMIZATION LAB

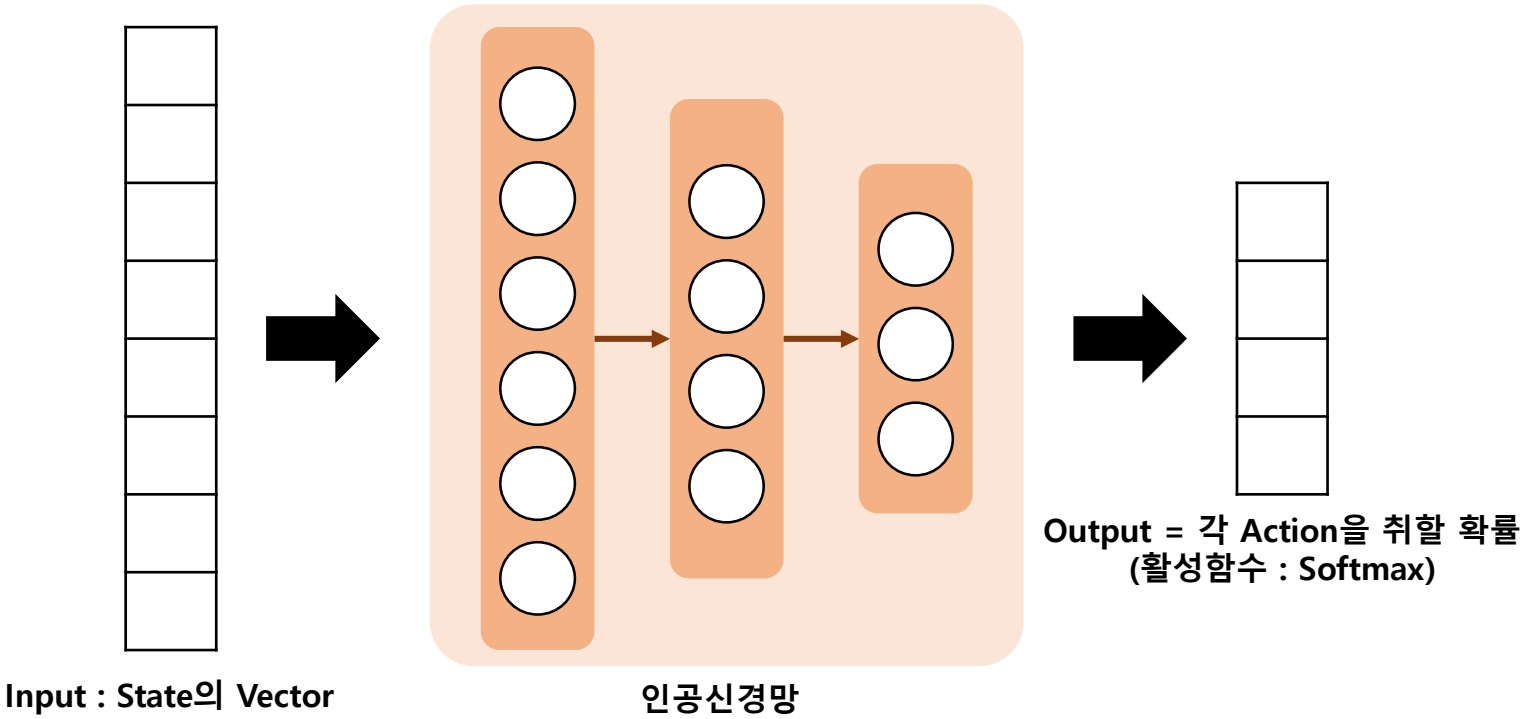
김 태 민

INU 인천대학교
INCHEON NATIONAL UNIVERSITY

Contents

Policy Gradient – Policy-based Reinforcement Learning

- **Policy-based Reinforcement Learning**
  - Policy-based Reinforcement Learning은 가치함수를 토대로 행동을 선택하지 않고, 상태에 따라 바로 행동을 선택한다.



  - Policy-based Reinforcement Learning은 위 그림에서의 인공신경망이 정책을 근사합니다.
  - 따라서 인공신경망의 입력은 상태가 되고 출력은 각 행동을 할 확률이 됩니다.

Policy Gradient – Policy Gradient

- **Policy Gradient**
  - 강화학습의 목적은 누적 보상을 최대로 하는 Optimal Policy를 찾는 것이다.
  - Policy Gradient는 π라고 표현되는 Policy를 직접적으로 모델링하고 최적화하는데 주력한다. 이 때, π는 정책 신경망으로 찾기 때문에 θ라는 정책 신경망의 가중치 값으로 표현된다. **(Policy = $\pi_\theta(a|s)$)**

  - Policy Gradient의 목적함수(Reward Function)는 J(θ)로 표현되며 그 식은 아래와 같다.

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^{\pi}(s,a)$$

  - 위 목적함수를 최적화하는 방법은 **J(θ)를 미분해서 그 미분값에 따라 정책을 업데이트하는 것**

    → 여기서 $d^{\pi}(s)$는 s라는 상태에 에이전트가 있을 확률이다.

      → policy π_θ를 가진 상태의 Markov Chain에 대한 Stationary distribution(상태 분포)

  - 정책 기반 강화학습의 목표는 Maximize J(θ)이므로 경사가 올라가야 한다. **(경사 상승법 (Gradient Ascent))**

Policy Gradient – Policy Gradient

- **Policy Gradient**

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s,a)$$

  - 위 식을 미분하면 아래와 같은 식이 최종적으로 도출된다.

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

  - 위 식에서 나온 기댓값은 샘플링으로 대체할 수 있으며, 결국 에이전트가 정책신경망을 업데이트하기 위해 구해야 하는 식은 다음과 같다.

$$Q^{\pi}(s,a) \nabla_{\theta} \ln \pi_{\theta}(a|s)$$

  - 위 식을 구한다면 정책 신경망의 계수는 경사상승법에 의해 업데이트할 수 있다.

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} J(\theta) \approx \theta_t + \alpha [\nabla_{\theta} log \pi_{\theta}(a \mid s) q_{\pi}(s,a)]$$

Policy Gradient의 θ 업데이트 식

Policy Gradient – REINFORCE

- **Policy Gradient Algorithm - REINFORCE**

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \approx \theta_t + \alpha [\nabla_\theta log \pi_\theta (a \mid s) q_\pi (s, a)]$$

Policy Gradient의 θ 업데이트 식

- 위 식에서 문제가 발생하는데, 현재 에이전트는 정책만 가지고 있지 Q함수를 가지고 있지 않다는 것이다.
- 이를 해결하기 위해 Q함수를 근사할 수 있는 반환값으로 대체한다.

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \approx \theta_t + \alpha [\nabla_\theta log \pi_\theta (a \mid s) G_t]$$

Episode가 끝날 때까지 기다려서 Episode동안 지나온 상태에 대한 반환값

만약, Episode가 6번만에 끝난다면 G값은?

$$G_5 = R_6$$
$$G_4 = R_5 + \gamma G_5$$
$$G_3 = R_4 + \gamma G_4$$
$$G_2 = R_3 + \gamma G_3$$
$$G_1 = R_2 + \gamma G_2$$

Policy Gradient – REINFORCE

- **Policy Gradient Algorithm – REINFORCE**
  - **REINFORCE 알고리즘**

Repeat {

1. $\pi_\theta(a|s)$로부터 샘플 궤적 τ = {s(0), a(0), s(1), a(1), …, s(T), a(T)}를 생성
2. Episode에서 반환값 G를 계산
3. Episode에서 Loss function 계산

$$\text{loss} = -\sum_{t=0}^{T}(\log \pi_\theta(a_t|s_t)G_t)$$

4. 정책 파라미터를 업데이트한다. (= 정책을 업데이트한다.)

$$\theta \leftarrow \theta + \alpha\nabla_\theta J_{(\theta)}$$

}

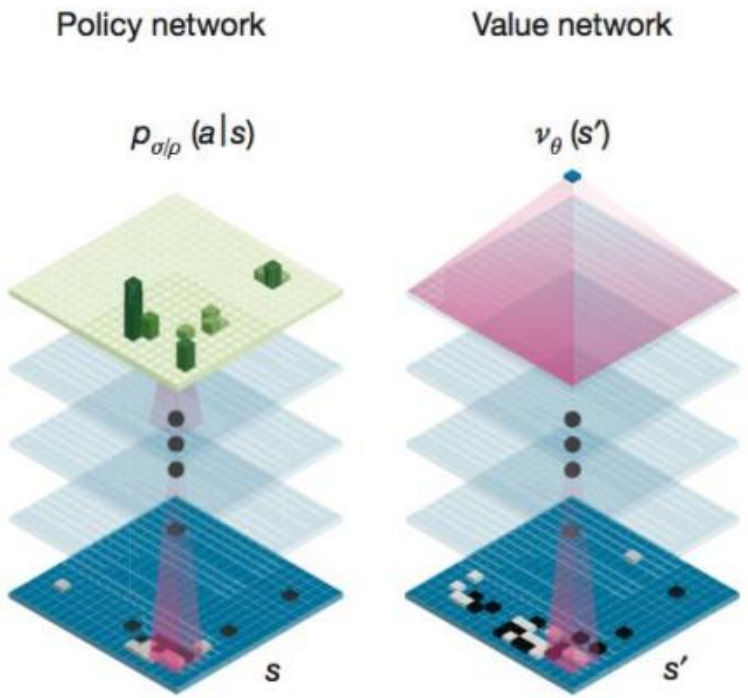
  - **REINFORCE 알고리즘의 단점**
    - 하나의 **Episode가 끝나야 Policy 업데이트가 가능**하다.
    - Gradient의 분산이 매우 크다. (반환값**(G)**이 Episode마다 크게 차이가 남에 따라 **목적함수의 Gradient도 값의 크기가 불안정**)
    - On-Policy 방법이다.

→ **On-Policy 방법은 정책 업데이트를 위해 해당 정책을 실행시켜 발생한 샘플이 필요함**

A2C

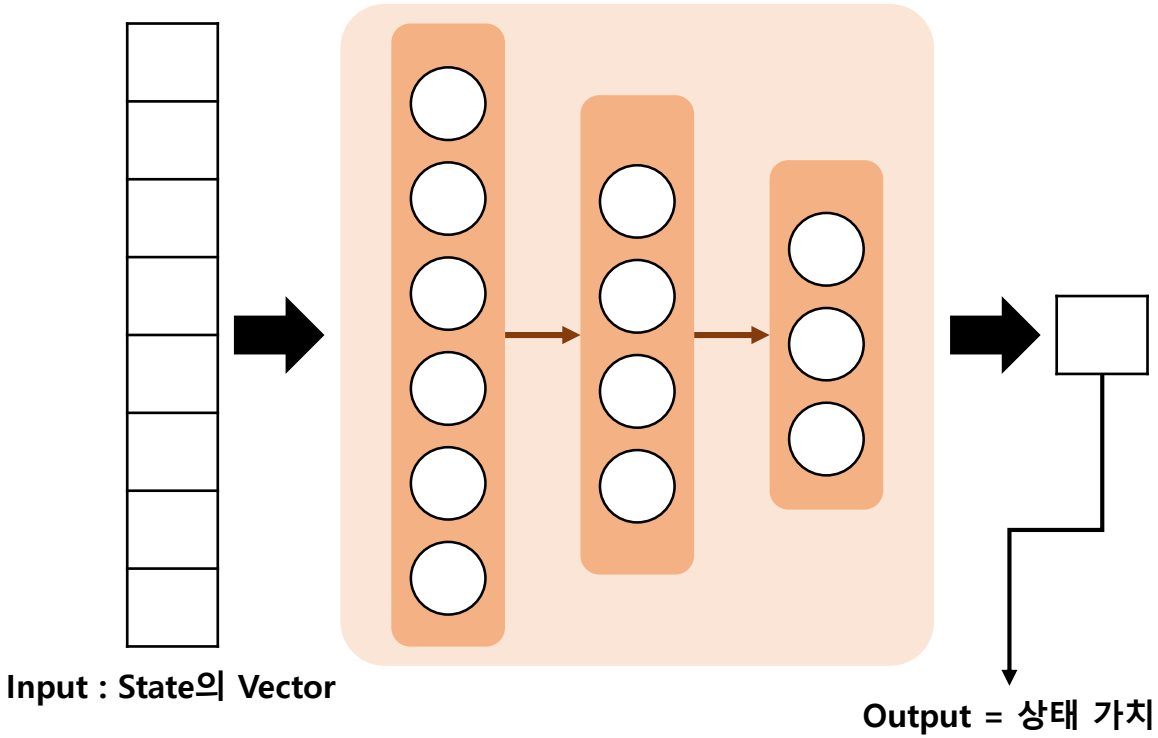
- **A2C의 탄생**
  - REINFORCE 알고리즘의 단점인, Episode가 끝날 때까지 기다려야 하고, Gradient의 분산이 매우 크다는 단점을 개선

- **Actor – Critic**
  - Value Network와 Policy Network를 결합
    - Value Network
      - DQN과 동일하게 가치를 학습
      - State와 Action에 대한 가치 도출
    - Policy Network
      - Policy gradient를 사용
      - 학습한 Policy를 이용하여 Agent의 의사결정 수행

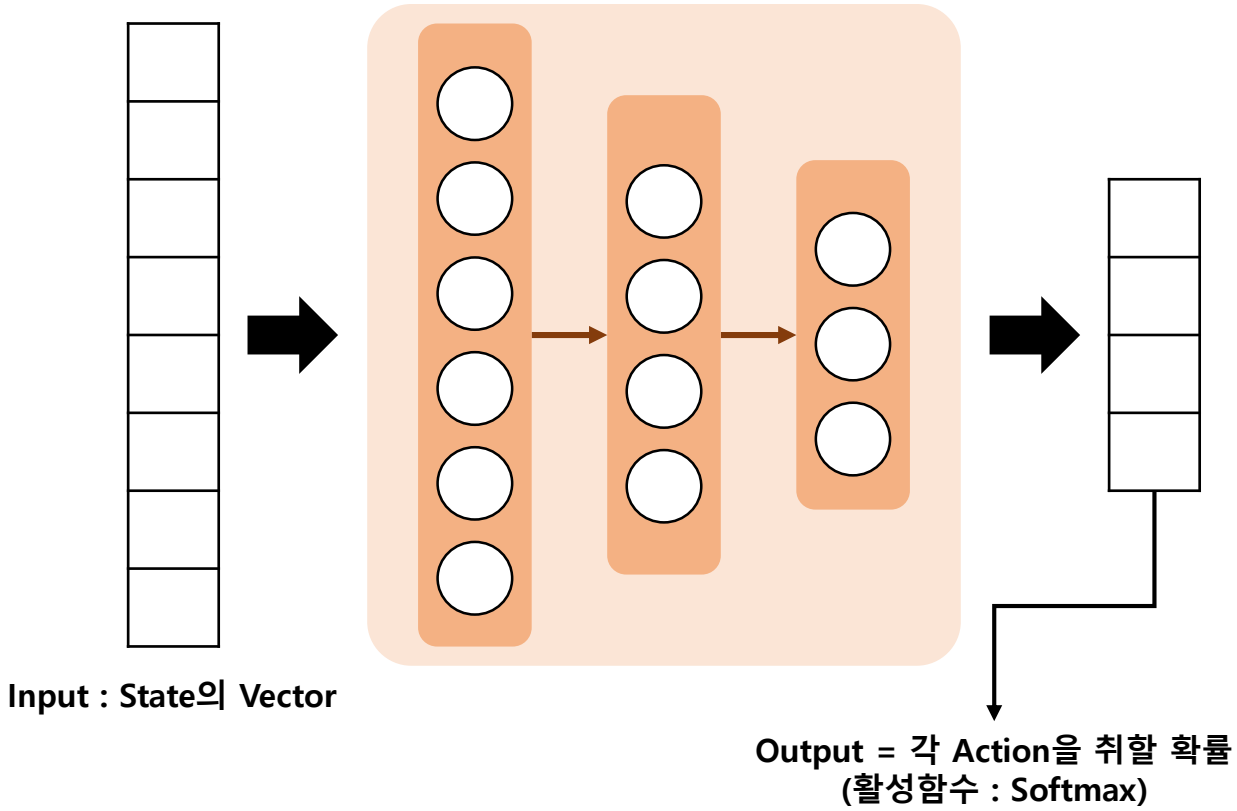


Deepmind의 AlphaGo의 네트워크

- Value Network



- Policy Network



A2C

- **A2C의 목적함수**

baseline

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) \cdot (q_\pi(s,a) - v_\pi(s))] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) A(s,a))]$$

- $A_\pi = q_\pi(s,a) - v_\pi(s) \rightarrow$ 어드밴티지 함수

A2C의 목적함수

- 가치 함수는 상태마다 값이 다르지만 행동마다 다르지는 않기 때문에 Q함수의 분산을 줄일 수 있다.

- **A2C의 손실함수**

  - **Policy Network**

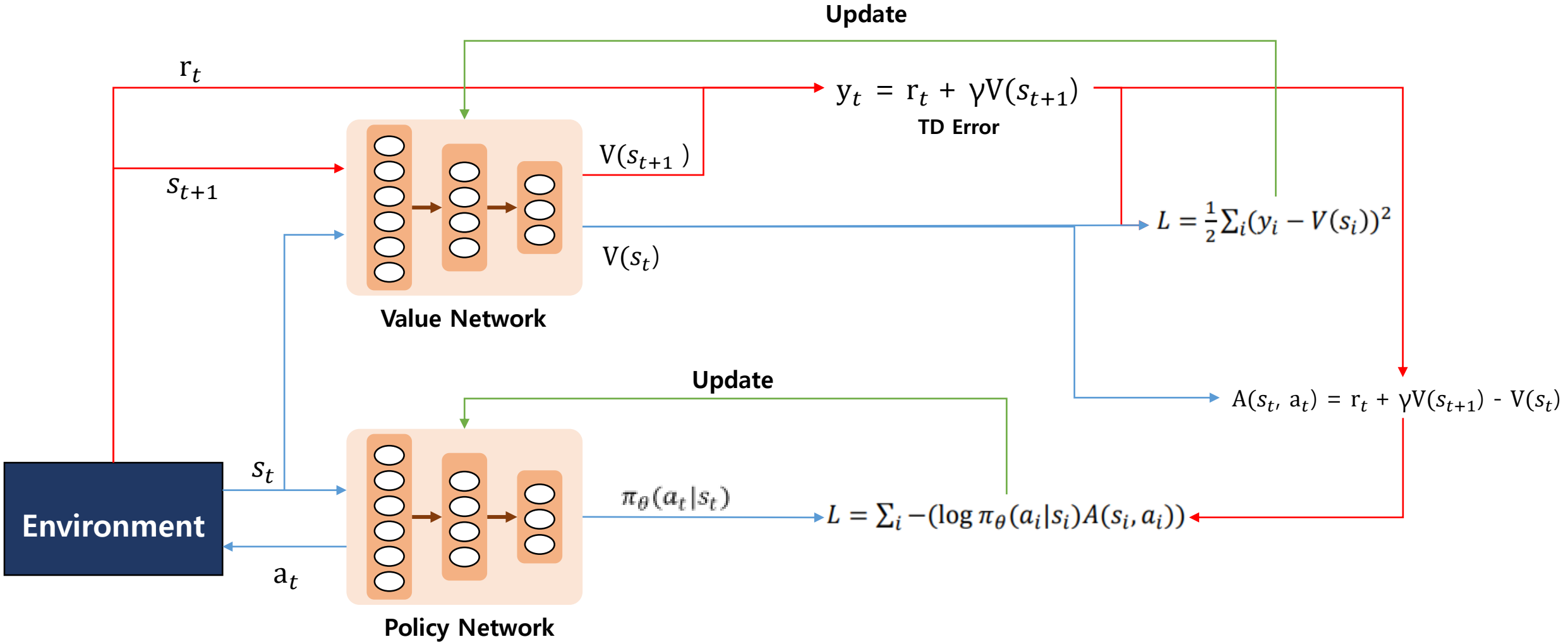
$$L = \sum_i -(\log \pi_\theta(a_i|s_i) A(s_i, a_i))$$

  - **Value Network**

$$L = \frac{1}{2}\sum_i (y_i - V(s_i))^2 \quad \text{where} \quad y_i = r_i + \gamma V(s_{i+1})$$

A2C

- A2C의 구조


Update
r_t
$y_t = r_t + \gamma V(s_{t+1})$
TD Error
$V(s_{t+1})$
s_{t+1}
$L = \frac{1}{2}\sum_i (y_i - V(s_i))^2$
$V(s_t)$
Value Network
Update
$A(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$
s_t
$\pi_\theta(a_t|s_t)$
$L = \sum_i -(\log \pi_\theta(a_i|s_i) A(s_i, a_i))$
Environment
a_t
Policy Network

DDPG 개요

- **Deep Deterministic Policy Gradient (DDPG)**
  - 기존 DQN 알고리즘은 이산적인 행동환경에만 적용 가능하기 때문에 연속적인 행동이 필요한 환경에서는 적용할 수 없다.

  - **DDPG**
    - Actor-Critic 기반 강화학습 알고리즘
    - DPG(Deterministic Policy Gradient) 알고리즘에 심층인공신경망 기법을 적용
    - 연속적인 행동을 위한 환경에 적용된 초기 심층강화학습 알고리즘
    - 성능이 불안정하고 어려운 환경에서는 학습이 잘 안되는 경우가 많음

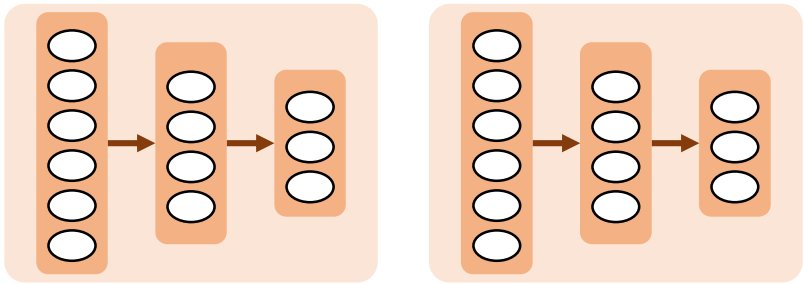
    - **이후 개선된 알고리즘**

      → TRPO, PPO, TD3 등이 존재

DDPG 기법

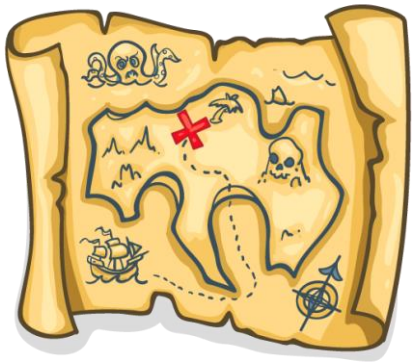
- DDPG에 쓰인 기법

경험 리플레이 (Experience Replay)



타겟 네트워크 (Target Network)

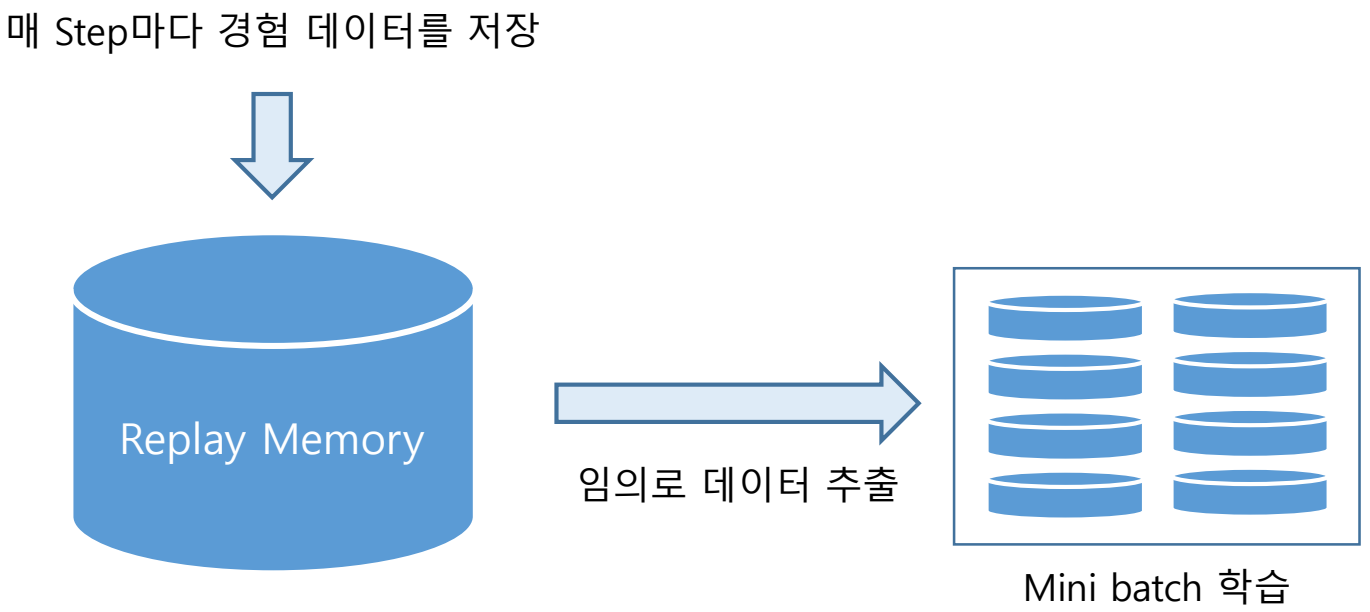
Soft Target Update 기법 사용



OU Noise를 사용한 탐험

DDPG 기법 – 경험 리플레이 (Experience Replay)

- **경험 리플레이 (Experience Replay)**
  - DQN에서 사용된 것과 동일한 기법 사용
  - 연속적인 결정 과정에서 발생하는 샘플 간의 상관관계 문제를 해결



DDPG 기법 – 타겟 네트워크 (Target Network)

- **타겟 네트워크 (Target Network)**
  - DQN과 같이 Target Network를 따로 설정하여 학습의 안정성 증가
  - DQN과 달리 매 Step Soft Target Update를 이용하여 Target Network 업데이트

  - **Soft Target Update**
    - 지수이동평균과 같은 방법을 통해 업데이트
    - 급격한 네트워크의 변화를 방지하여 안정적으로 네트워크가 수렴하도록 도와준다.

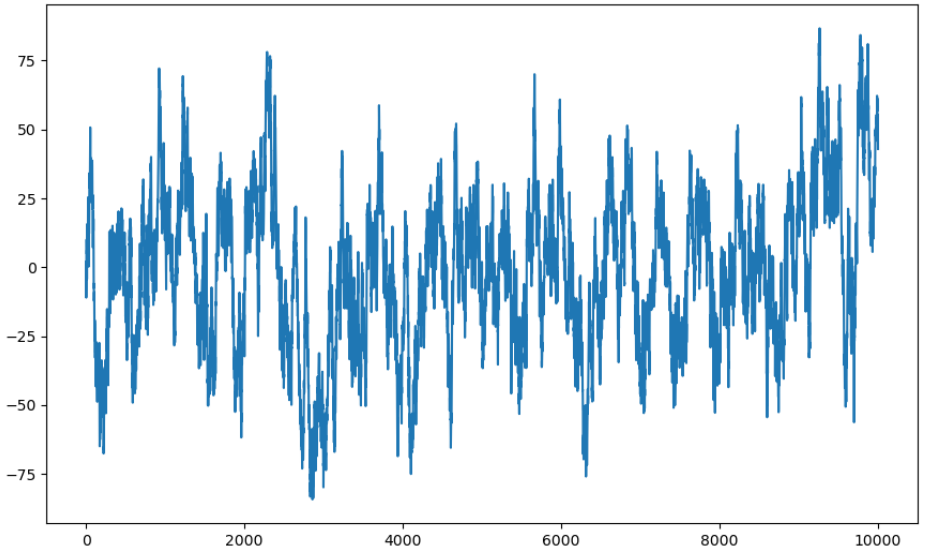
$$\theta^{-}_{critic} \leftarrow \tau\theta_{critic} + (1-\tau)\theta^{-}_{critic}$$

$$\theta^{-}_{actor} \leftarrow \tau\theta_{actor} + (1-\tau)\theta^{-}_{actor}$$

τ → 업데이트 비율

DDPG 기법 – OU Noise (Ornstein Uhlenbeck Noise)

- **OU Noise**
  - 연속 행동 환경에서는 행동의 경우의 수가 무한대이기 때문에 기존의 e-greedy 기법을 쓸 수 없다.
  - 실수 범위에서 행동을 선택하여 탐험할 수 있는 랜덤 평균 회귀 노이즈를 생성할 필요가 있다.



OU Noise 그래프

DDPG에서의 Action

$$\text{Action} = u(s_t) + \text{N}$$

$u(s_t)$: Actor Network를 통해 결정된 값

N : Noise 값

DDPG 학습 – Critic Network Update

- **Critic Network Update**
  - Q함수 값에 대한 예측값과 Target Value의 차이를 줄이는 방향으로 업데이트

$$y\,(target\ value) = \begin{cases} r & \text{다음 Step에서 게임이 종료될 때} \\ r + \gamma \left(\max_{a'} Q\,(s_{t+1}, a'; \theta^{-}) \right) & \text{게임이 계속 진행되는 경우} \end{cases}$$

  - Loss Function의 경우 예측값과 Target Value의 차이가 제곱 평균인 MSE로 설정

$$\text{Loss} = \frac{1}{N} \sum_{t} (Q(s_t, a_t; \theta) - y_t)^2$$

Loss 값을 최소화하는 방향으로 학습 진행

DDPG 학습 – Actor Network Update

- **Actor Network Update**
  - 목표함수를 최대화하는 방향으로 정책을 업데이트

목표함수 $J = v_{\pi_\theta}(s_0)$

$= v_{\mu_\theta}(s_0)$ ⟶ 행동들의 확률분포를 출력하는 것이 아닌 한 가지 행동만 출력하는 정책 (μ)

$$= \sum_t \left[\rho^{\mu}(s) Q(s, a; \theta^Q) \big|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right]$$ ⟶ 큐함수를 이용한 변형

θ^μ: 액터 네트워크의 파라미터
θ^Q: 크리틱 네트워크의 파라미터
$\rho^\mu(s)$: 정책 μ를 통해 에피소드를 진행했을 때 상태 s를 방문할 확률

DDPG 학습 – Actor Network Update

- **Actor Network Update**
  - 목표함수를 최대화하는 방향 계산을 위한 Gradient 구하기

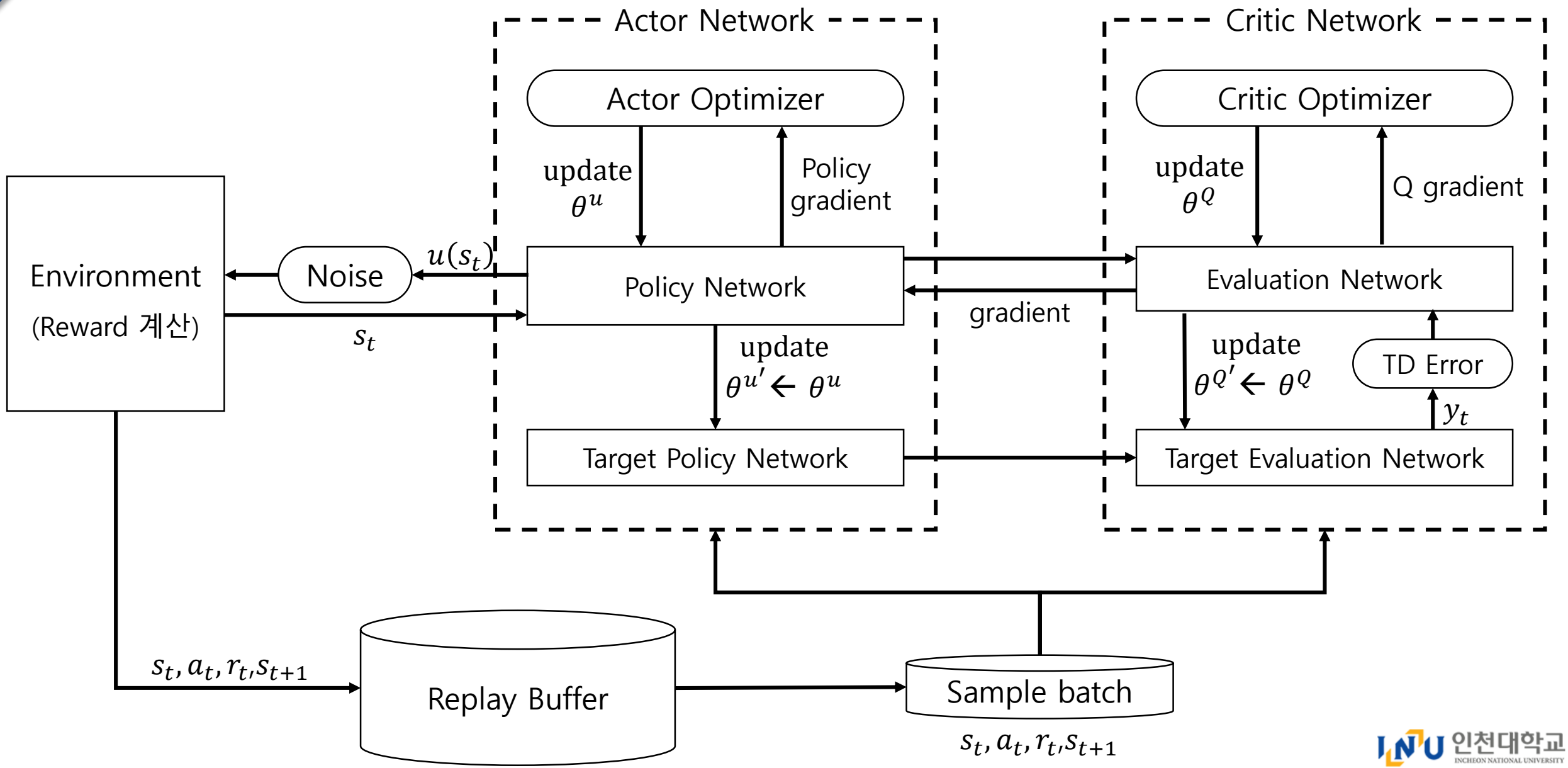
$$\nabla_{\theta^\mu} J = \sum_t \left[\rho^\mu(s) \nabla_{\theta^\mu} Q(s,a;\theta^Q) \big|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right]$$

$$\nabla_{\theta^\mu} J = \sum_t \left[\rho^\mu(s) \nabla_a Q(s,a;\theta^Q) \big|_{s=s_t, a=\mu(s_t|\theta^\mu)} \nabla_{\theta^\mu} \mu(s;\theta^\mu) \big|_{s=s_t} \right]$$

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_t \left[\nabla_a Q(s,a;\theta^Q) \big|_{s=s_t, a=\mu(s_t|\theta^\mu)} \nabla_{\theta^\mu} \mu(s;\theta^\mu) \big|_{s=s_t} \right]$$

상태방문확률을 알기 어렵기 때문에 총 Step인 N으로 나눠 근사값을 구함

DDPG 구조



참고문헌

- https://talkingaboutme.tistory.com/entry/RL-Policy-Gradient-Algorithms
- https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html
- Policy Gradient Methods for Reinforcement Learning with Function Approximation, Richard S. Sutton 외 3명, Advances in Neural Information Processing Systems 12 (NIPS 1999)
- https://zoomkoding.github.io/%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5/2019/08/07/RL-5.html
- https://wnthqmffhrm.tistory.com/19
- Continuous control with deep reinforcement learning, Timothy P. Lillicrap 외 7명, arXiv preprint arXiv:1509.02971 (2015)
- http://wiki.hash.kr/index.php/DDPG
- 파이썬과 케라스로 배우는 강화학습, 위키북스
- 수학으로 풀어보는 강화학습 원리와 알고리즘, 위키북스
- 강화학습 / 심층강화학습 특강, 위키북스

감사합니다