DDPG tutorial

김경환

medipixel

발표자 소개

김경환 Kim Kyunghwan

kh.kim@medipixel.io / khsyee@gmail.com

AI Research / Developer at Medipixel

Contributor at ModuLabs

RL Lecturer at {Multicampus, LG, ...}

Hansung University
(Majored in Electronic Information Engineering)

Interests : AI, Reinforcement Learning, (Playing games!)

medipixel

목 차

1. Review
2. Deterministic PG
3. Deep Deterministic PG
4. Practice

Review

Policy Search

- ❏ Value Function approach
  - 파라미터로 Value function을 근사
  - Value function을 최대화하는 action을 선택
- ❏ Policy Search
  - 파라미터로 Policy를 근사
  - Policy (확률분포)로부터 action을 sampling

Policy Gradient

- Performance measure

  - Policy를 reward를 통해 평가

$$
\begin{aligned}
J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) r(s,a) da ds \\
&= E_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)]
\end{aligned}
$$

  - Performance measure가 최대가 되도록 파라미터 조정

$$\theta = \text{maximize } J(\pi_\theta)$$

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\pi_\theta)$$

Policy Gradient

- Policy Gradient

  - Policy gradient theorem

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(s,a)\ Q^\pi(s,a)\ dads \\
&= E_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\ Q^\pi(s,a)]
\end{aligned}
$$

PG with Function Approximation

- Function Approximation Q

$$Q^w(s,a) \approx Q^\pi(s,a)$$

- Minimize MSE error

$$\epsilon^2(w) = E_{s\sim\rho^\pi, a\sim\pi_\theta}[(Q^w(s,a) - Q^\pi(s,a))^2]$$

Deterministic PG

Deterministic Policy

- Stochastic vs Deterministic

$$\pi_\theta(s, a) = P[a|s; \theta]$$

vs

$$\mu_\theta(s) = a$$

Deterministic Policy

- 차이점
  - Input argument
    - Stochastic : State, Action space에 대해서 고려해야 함.
    - Deterministic : State space만 고려함.
  - Exploration
    - Stochastic : action의 확률 분포를 출력, exploration 효과가 있음.
    - Deterministic : 하나의 action이 출력되기 때문에 exploration이 따로 필요함.

Stochastic & Deterministic Policy의 관계

- Stochastic policy parameterization

$$\pi_{\mu_\theta,\sigma}$$

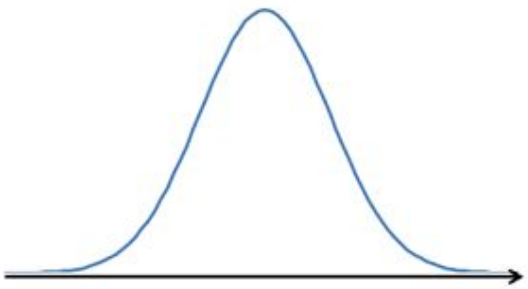
Deterministic policy: $\mu_\theta : S \rightarrow A$

Variance parameter: σ

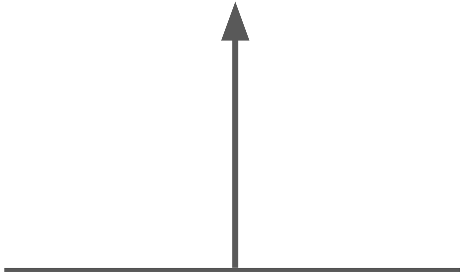
Stochastic & Deterministic Policy의 관계

❑ Stochastic policy와 Deterministic policy

$$\pi_{\mu_\theta,0} \equiv \mu_\theta$$



$\pi_{\mu_\theta,\sigma}$



μ_θ

Deterministic PG

- Performance Measure

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) r(s,a) dads \\ &= E_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)] \end{aligned}$$

$$\begin{aligned} J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) r(s, \mu_\theta(s) ds \\ &= E_{s\sim\rho^\mu}[r(s, \mu_\theta(s))] \end{aligned}$$

Deterministic PG

- Deterministic Policy Gradient

$$
\begin{aligned}
\nabla_\theta J(\mu_\theta) &= E_{s\sim\rho^\mu}\left[r(s,\mu_\theta(s))\right] \\
&= \nabla_\theta Q^\mu(s,\mu_\theta(s)) \\
&= \int_S \rho^\mu \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} \\
&= E_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]
\end{aligned}
$$

appendix 참조: http://proceedings.mlr.press/v32/silver14-supp.pdf

SPG와 DPG의 관계

❑ DPG: Special case of the SPG

$$\lim_{\sigma \downarrow 0} \nabla_\theta J(\pi_{\mu_\theta, \sigma}) = \nabla_\theta J(\mu_\theta)$$

∴ Deterministic PG를 기존 policy gradient 기법들에 적용 가능

e.g. PG with function approximation, on-policy / off-policy actor critic, ...

Deterministic Actor Critic

- Function Approximation Q

$$Q^w \approx Q^\mu$$

- Minimize MSE error

$$\epsilon^2(w) = E_{s\sim\rho^\mu}[(Q^\mu(s,a) - Q^w(s,a))^2]$$

Deterministic Actor Critic

- Off-Policy Deterministic Actor Critic

  - Critic : Q-learning

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t))$$

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t)$$

  - Actor: Deterministic PG

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)}$$

Deep Deterministic PG

DQN

❑ DQN

- DQN은 DNN과 강화학습을 결합하여 좋은 성능을 보여줌.
- **But,** Discrete 또는 Low dimensional action space에만 적용 가능.

DQN의 장점은 가지고 있고
Continuous action space에 적용할 방법이 없을까?

DQN +

DQN **+**

DQN + DPG

DQN **+** **DPG**

DDPG

DQN + **DPG**

= **DDPG !**

DDPG

❏ DQN

- Experience Replay, Target Network 기법을 그대로 사용
- Critic network가 DQN network와 동일

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t))$$

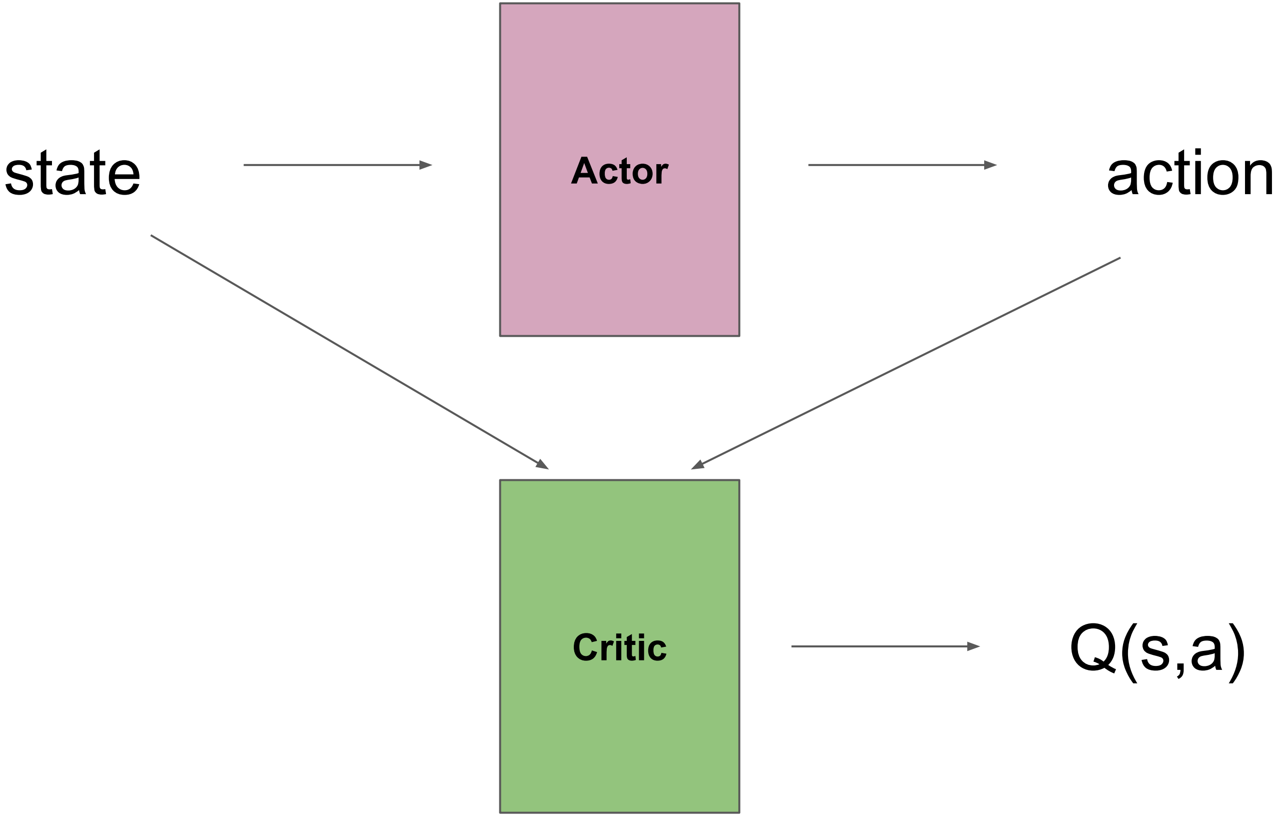
$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t)$$

DDPG

- DPG
  - Actor network를 DPG로 업데이트

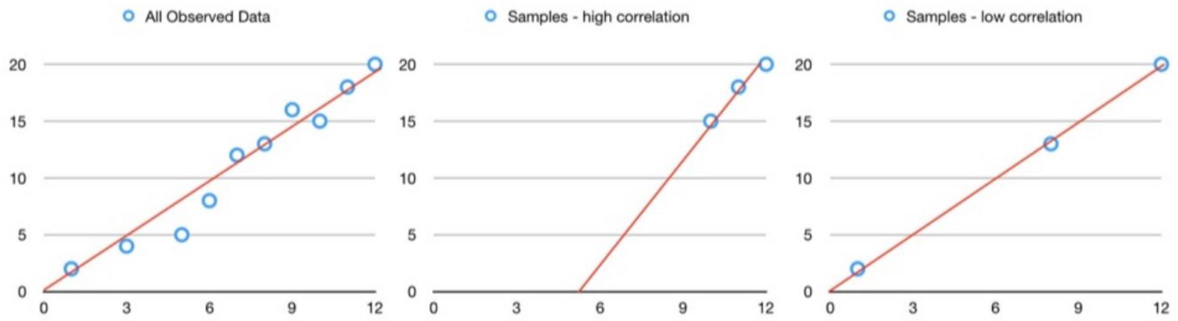
$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)}$$

DDPG



Experience Replay

- ❑ Correlation between samples
  - 강화학습에서의 sample은 시간에 따라 순차적으로 수집되기 때문에 correlation이 높다
  - Sample간의 correlation이 높으면 학습이 불안정해진다.



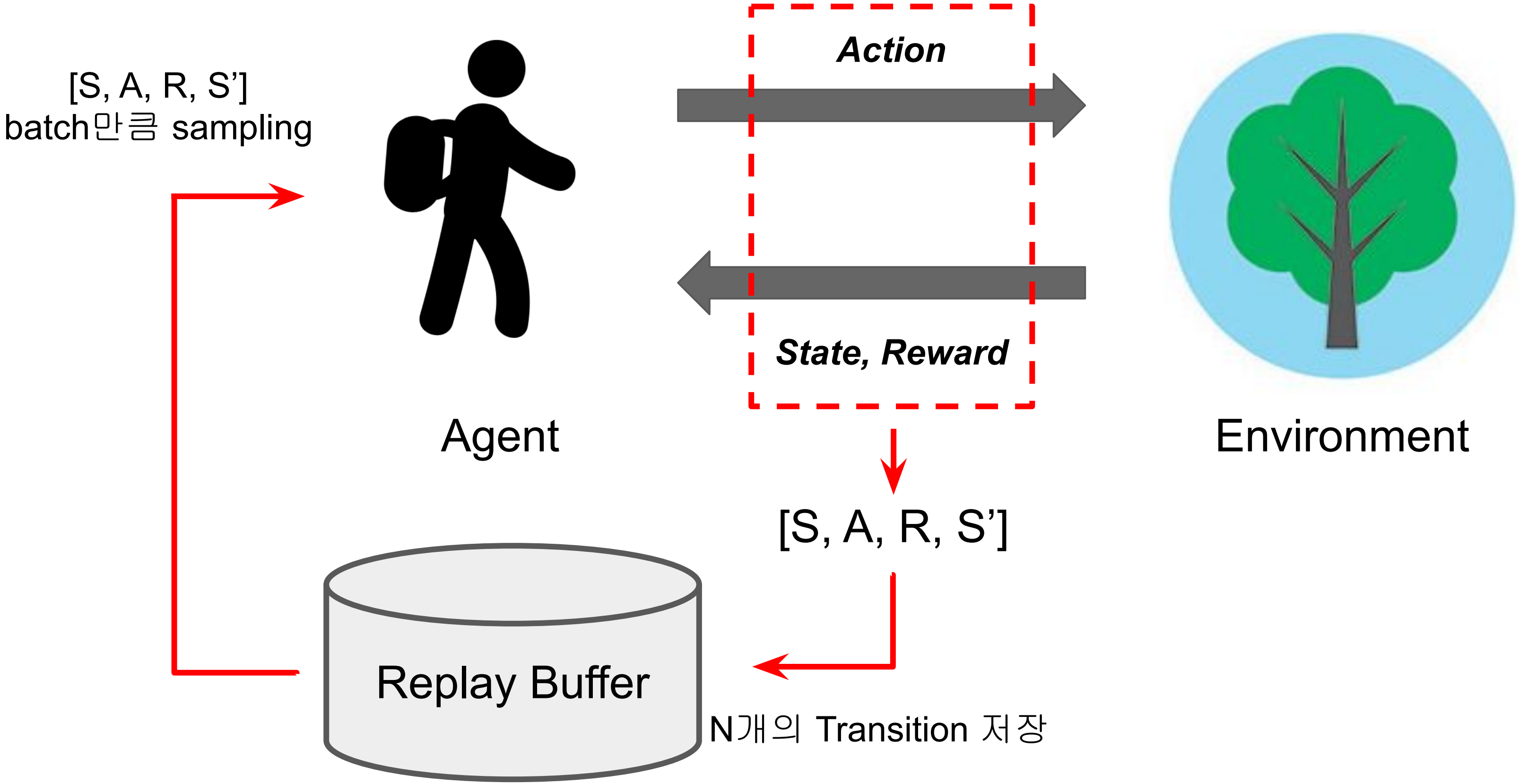
ref: 강화학습 기초부터 DQN까지 (박진우)
https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

Experience Replay

❑ Experience replay

- transition(S, A, R, S’)을 memory(buffer)에 저장하고 batch 단위로 학습하자.
- data(transition)간의 correlation을 없앰.
- batch 단위로 학습 가능.

Experience Replay



Target Network

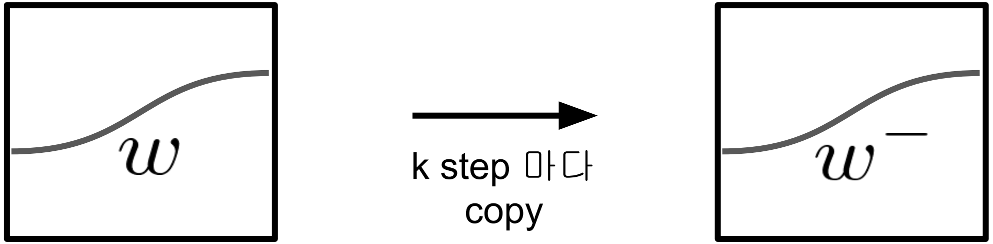
❑ Non-stationary targets

$$\text{Loss} = (\underbrace{R + \gamma \max_{a} Q(S', a; w)}_{\textit{target}} - Q(S, A; w))^2$$

- Loss function에서 target 과 current value 가 모두 파라미터 w를 통해 계산됨.
- w가 업데이트 되면 target도 바뀌어 버림.

Target Network

- Target network
  - 일정 step마다 업데이트 되는 network를 추가하여 update시의 target으로 사용



Main Network *Target Network*

$$\text{Loss} = (R + \gamma \max_a Q(S', a; w) - Q(S, A; w))^2$$

$$\text{Loss} = (R + \gamma \max_a Q(S', a; w^-) - Q(S, A; w))^2$$

Target Network

- DDPG Target network

$$\text{Loss} = (R + \gamma \max_a Q(S', a; w^-) - Q(S, A; w))^2$$

$$\downarrow$$

$$\text{Loss} = (R + \gamma \underline{Q(S', \mu(S'; \theta^-)}; w^-) - Q(S, A; w))^2$$

Target Network

❑ Soft target update

- target network를 한번에 완전히 변경하는 것이 아니라
- 매 step마다 조금씩 변경함.
- 이를 통해 좀 더 stable하게 학습하도록 함.

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \quad with\ \tau \ll 1$$

OU noise

- ❑ Ornstein-Uhlenbeck noise

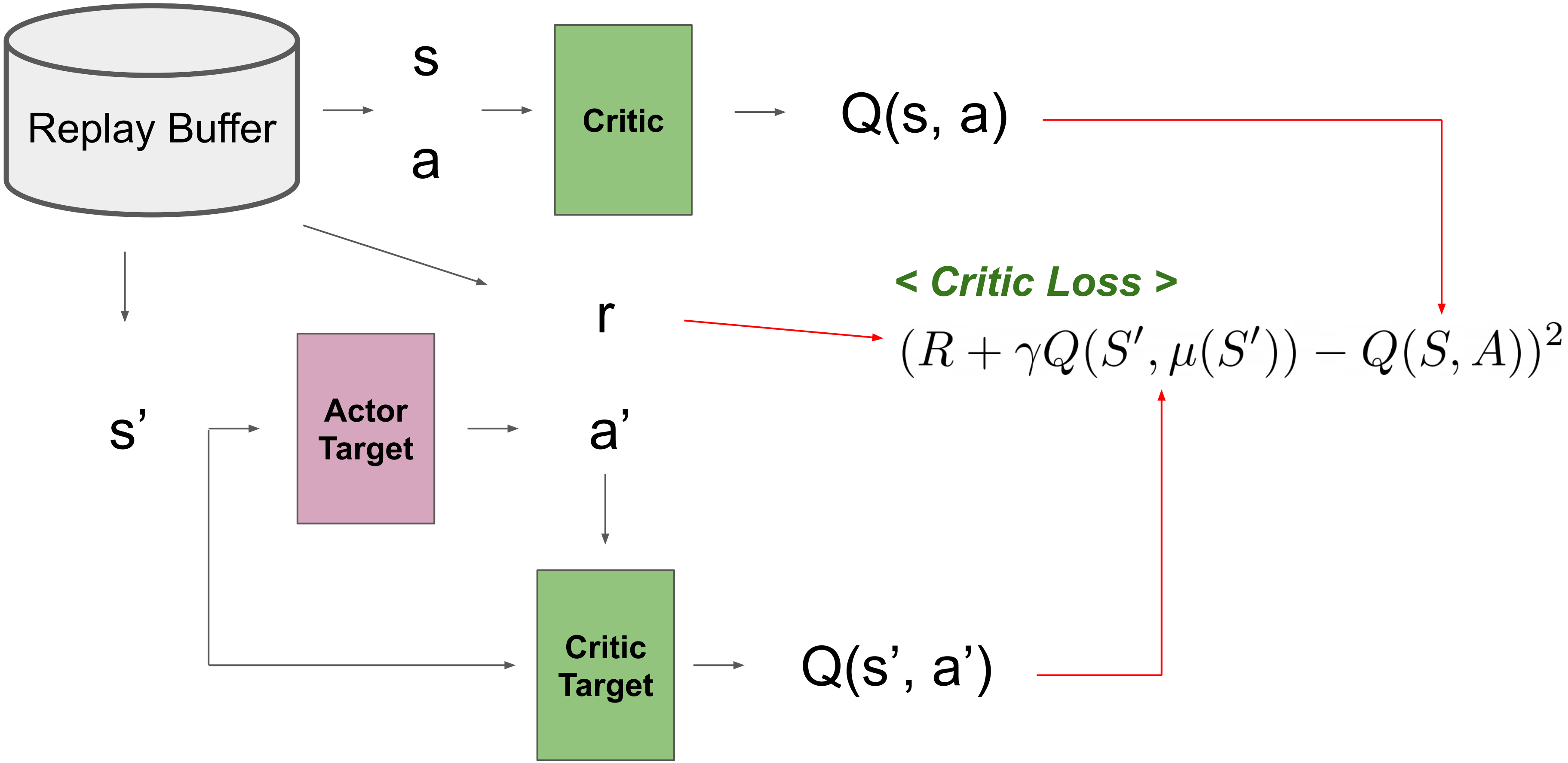
$$\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$$

- Exploration을 위해 action에 noise를 추가
- 논문에서는 **OU noise**를 제안함

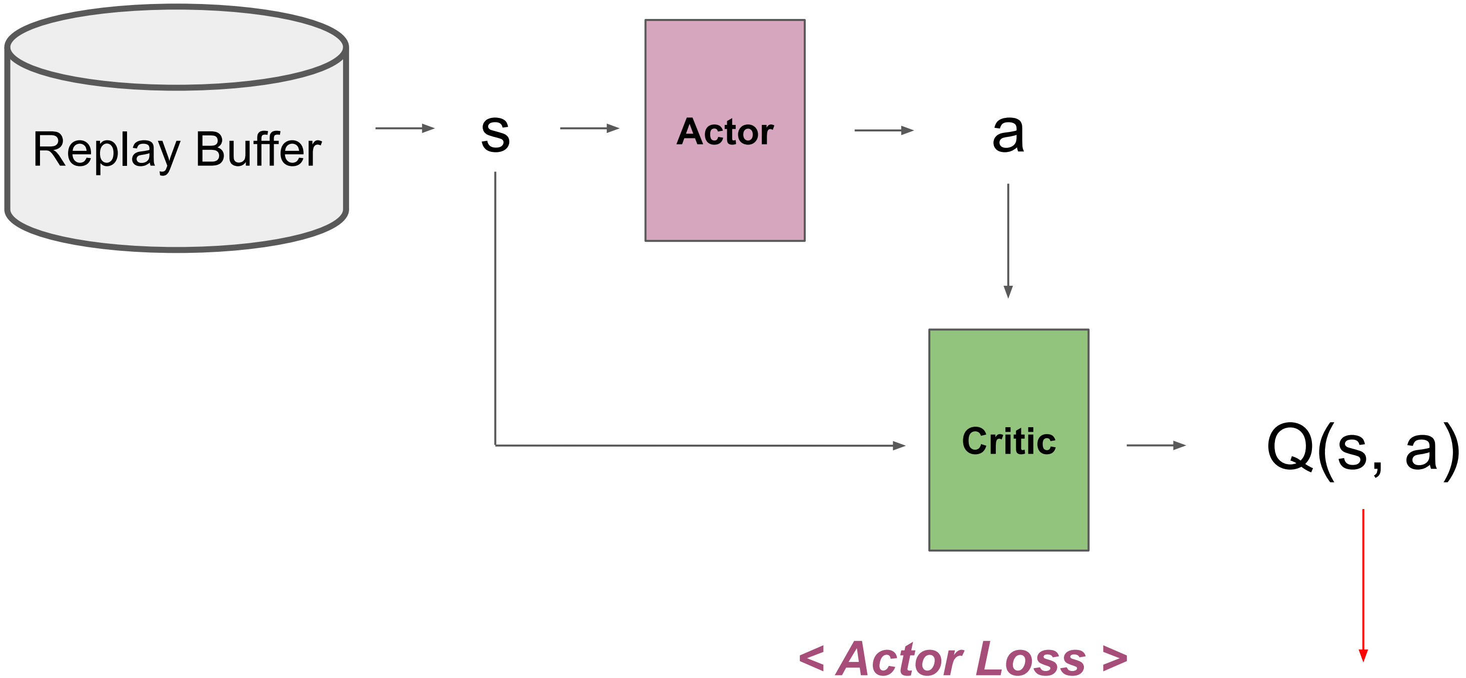
OU noise: $$dx_t = \theta(\mu - x_t)dt + \sigma dW_t$$

wiki: https://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process

DDPG - Critic update



DDPG - Actor update



< Actor Loss >

$$-Q^w(S, \mu_\theta(S))$$

Code

실습

https://github.com/MrSyee/pg-is-all-you-need