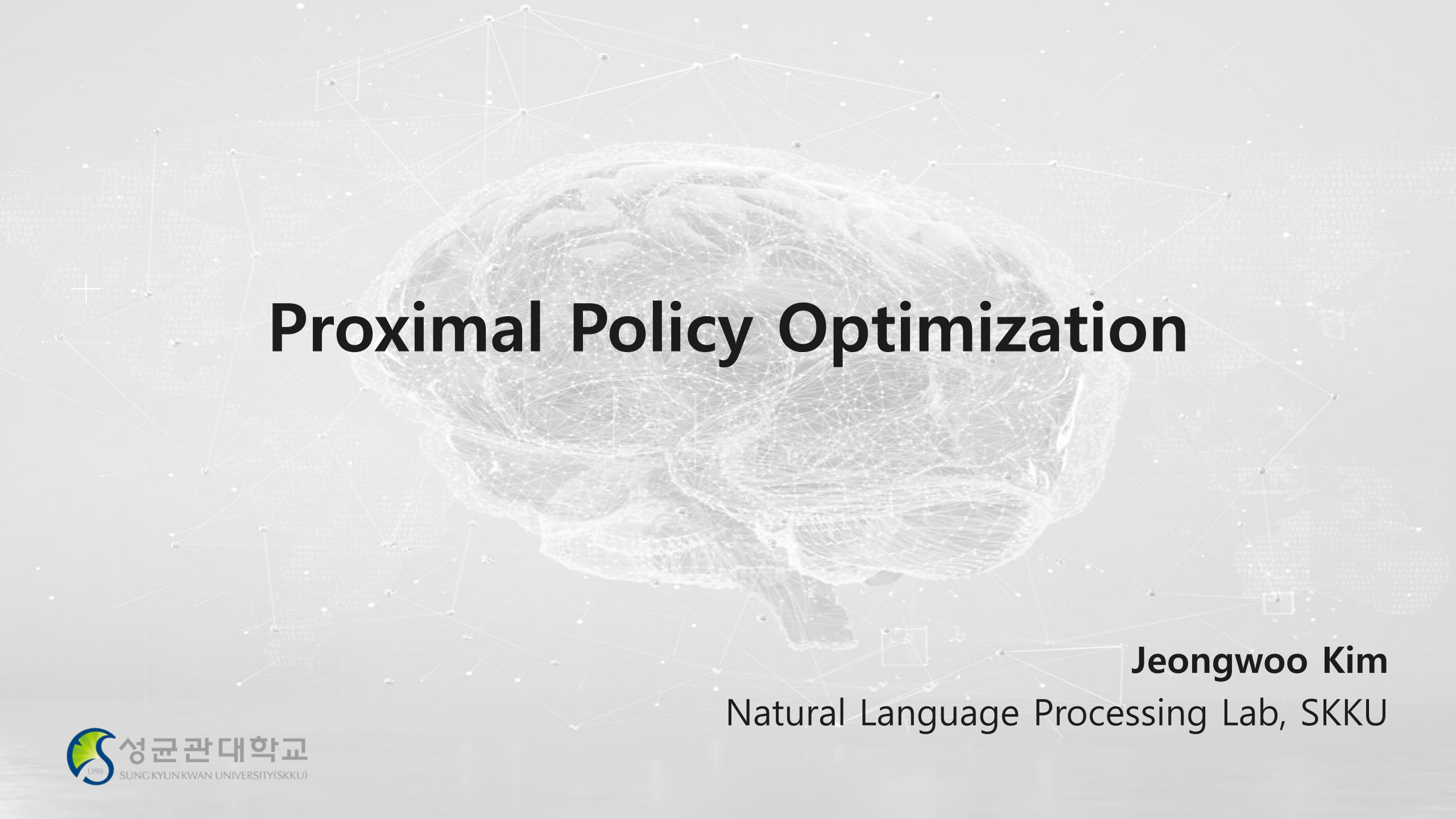


# Proximal Policy Optimization

**Jeongwoo Kim**
Natural Language Processing Lab, SKKU

성균관대학교
1398 SUNG KYUN KWAN UNIVERSITY(SKKU)

# Contents

1. Introduction

2. Preliminary

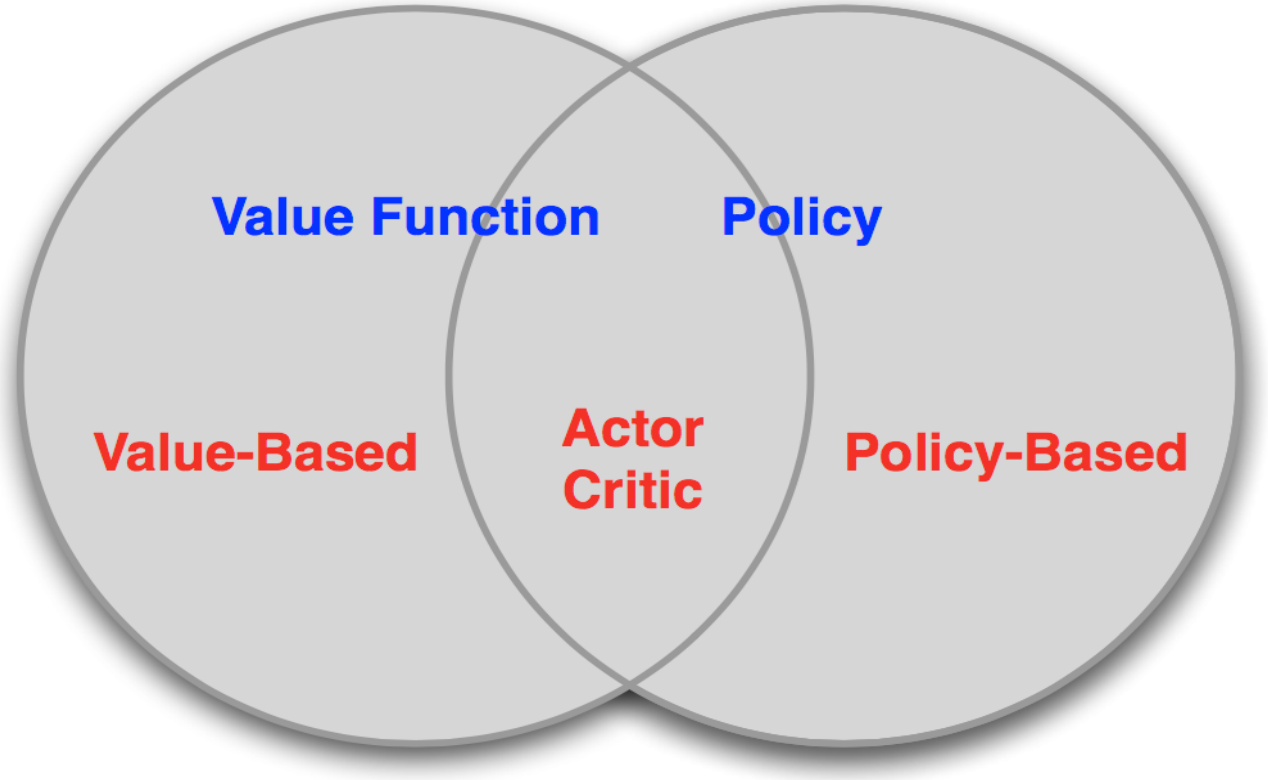
3. Method

4. Experiments

5. Conclusion

# Reinforcement Learning

## RL Algorithms

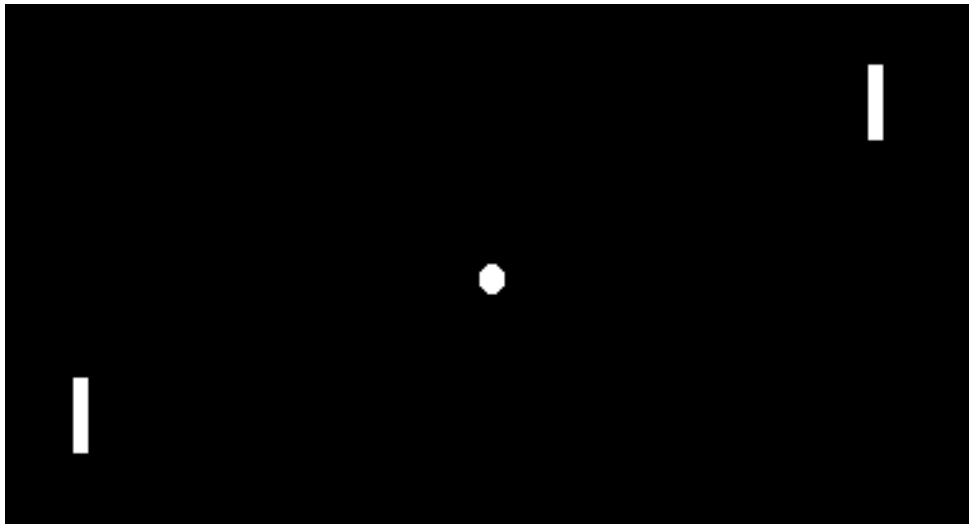


# Reinforcement Learning

- **RL Algorithms**
  - **Value-Based Methods**
    - 상태나 행동의 가치를 추정한 뒤, 가치가 가장 큰 행동을 선택
  - **Policy-Based Methods**
    - 정책 자체를 직접 학습하여, 기대 보상을 최대화
  - **Actor Critic**
    - 정책을 학습하는 actor와 가치를 추정하는 critic을 함께 사용해 더 안정적으로 정책 업데이트

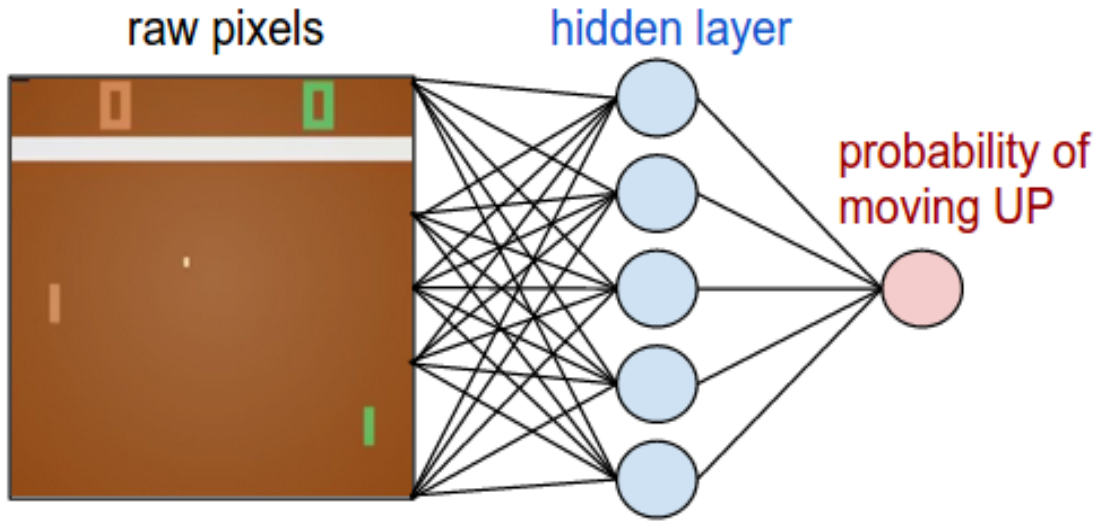
# Policy Gradient

- **Example: Pong as MDPs**
  - 입력(state): 게임 화면 픽셀
  - 행동(action): paddle을 UP/DOWN으로 이동
  - 보상(reward)
    - +1: 상대를 득점 못 하게 하고 내가 점수 얻음
    - -1: 공을 놓쳐 실점
    - 0: 그 외 대부분의 시간 step
  - 목표: 장기적으로 누적 보상이 최대가 되도록 정책 학습



# Policy Gradient

- **Policy Network**
  - Policy network는 현재 상태를 받아 행동 확률을 출력
  - 예제에서는 단순한 2-layer fully connected network 사용
  - 실제 행동은 argmax가 아니라 확률적으로 샘플링
  - 즉, policy는 deterministic하지 않고 stochastic policy로 구성됨

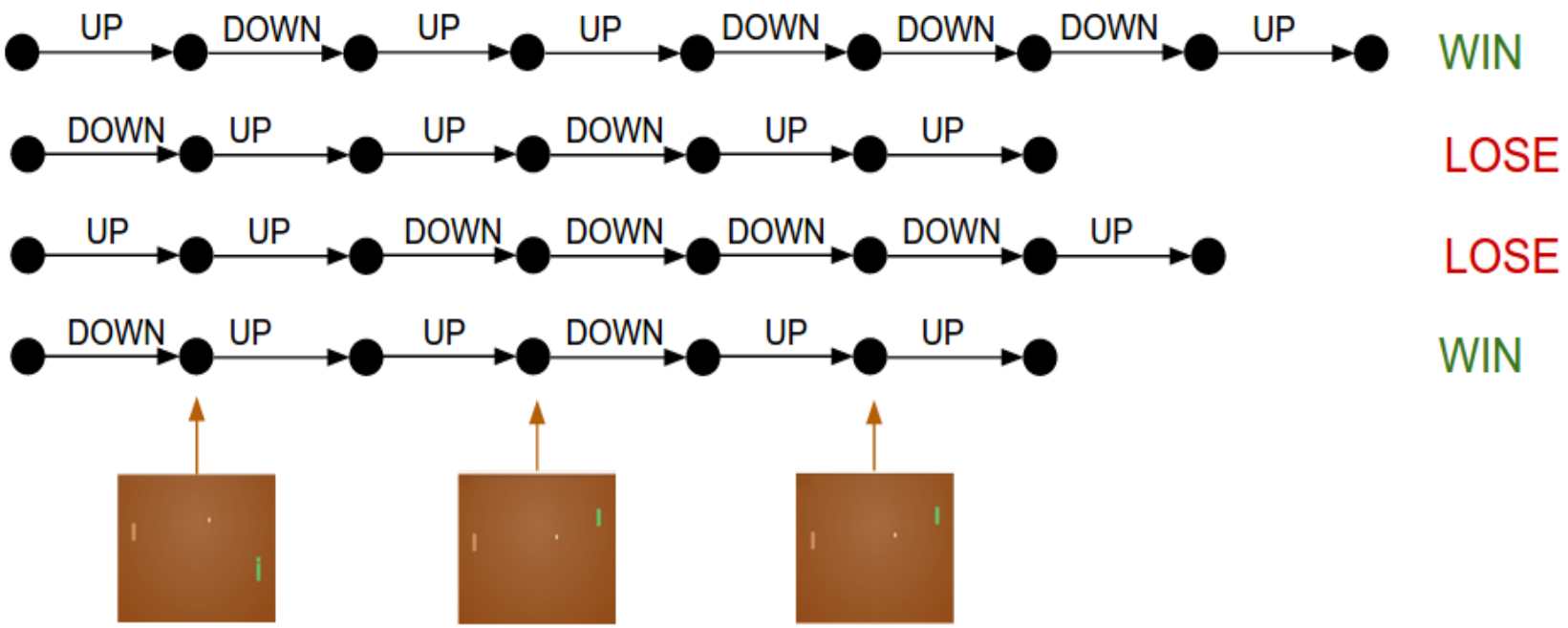


# Policy Gradient

## Objective

- A trajectory $\tau$ is a state, action, reward sequence returned by a policy.

$$\tau=(s_0, a_0, \mathrm{r}_0, .., s_T)$$



# Policy Gradient

## Objective

- A **trajectory** $\tau$ is a state, action, reward sequence returned by a policy.

$$\tau=(s_0, a_0, r_0, .., s_T)$$

- Let $R(\tau)$ give the **return** for the trajectory.

$$R(\tau)=\sum_{t=0}^{T-1} r_t$$

- We want to estimate this **gradient**:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$$

# Policy Gradient

## Objective

- We thinking of $\tau$ as a random variable and we want to do this

$$\nabla_\theta \mathrm{E}[R(\tau)] = E[\nabla_\theta P(\tau|\theta) R(\tau)]$$

- To do this, we need to expand and understand **policy** $P(\tau|\theta)$

$$P(\tau|\theta) = \mu(\mathrm{s}_0) \prod_{\mathrm{t}=0}^{\mathrm{T}-1} \pi_\theta(\mathrm{a}_\mathrm{t}|\mathrm{s}_\mathrm{t}) P(\mathrm{s}_{\mathrm{t}+1}, \mathrm{r}_\mathrm{t}|\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t})$$

초기 상태 확률

현재 policy가 각 시점의 행동을 선택할 확률

다음 상태와 보상이 주어질 확률

# Policy Gradient

## Objective

- We thinking of $\tau$ as a random variable and we want to do this

$$\nabla_\theta \mathrm{E}[R(\tau)] = E[\nabla_\theta P(\tau|\theta) R(\tau)]$$

- To do this, we need to expand and understand **policy** $P(\tau|\theta)$

$$P(\tau|\theta) = \mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) P(s_{t+1}, r_t|s_t, a_t)$$

$$\log(P(\tau|\theta)) = \log\mu(s_0 + \sum_{t=0}^{T} [\log\pi_\theta(a_t|s_t) + \log P(s_{t+1}, r_t|s_t, a_t)]$$

$$\nabla_\theta \log P(\tau|\theta) = \nabla_\theta \sum_{t=0}^{T-1} \log\pi_\theta(a_t|s_t)$$

# Policy Gradient

## Objective

- We thinking of $\tau$ as a random variable and we want to do this

$$\nabla_\theta \mathrm{E}[R(\tau)] = E[\nabla_\theta P(\tau|\theta) R(\tau)]$$

$$\nabla_\theta \mathrm{E}[R(\tau)] = E\left[\mathrm{R}(\tau)\nabla_\theta \sum_{t=0}^{T-1} \log\pi_\theta(a_t|s_t)\right]$$

$$= E\left[\sum_{t=0}^{T-1} r_t \nabla_\theta \sum_{t=0}^{T-1} \log\pi_\theta(a_t|s_t)\right]$$

$$= E\left[\nabla_\theta \sum_{t=0}^{T-1} \log\pi_\theta(a_t|s_t)\left(\sum_{t\prime=t}^{T-1} r_{t\prime}\right)\right]$$

sum of future reward

# Policy Gradient

## Summary

### Objective

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

### Policy Gradient Theorem

$$\nabla_\theta \mathrm{E}[R(\tau)] = E\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'}\right)\right]$$

# Policy Gradient

## From Return to Advantage

- 시점 $t$부터 미래에 받는 reward의 누적합을 **return** $G_t$로 나타낸다.

$$G_t = \sum_{t'=t}^{T-1} r_{t'}$$

- 시간이 멀어질수록 미래의 reward는 현재 행동의 영향과 덜 직접적으로 연결된다고 보고, $\gamma^{t'-t}$ 형태의 **discount factor**를 적용할 수 있다.

$$G_t = \sum_{t'=t}^{T-1} r^{t'-T} r_{t'}$$

# Policy Gradient

## From Return to Advantage

- $G_t$를 그대로 사용하면 샘플마다 변동이 커서 gradient 분산이 커질 수 있으므로, 상태 $s_t$에서의 평균적인 기대 return을 **baseline** $b(s_t)$로 둔다.

$$b(s_t) \approx V^{\pi}(s_t)$$

- 실제 return이 baseline보다 얼마나 더 좋은지를 **advantage** $A_t$로 정의한다.

$$A_t = G_t - b(s_t) \approx G_t - V^{\pi}(s_t)$$

# Policy Gradient

## From Return to Advantage

### Policy Gradient Theorem with Advantage

$$\nabla_\theta \mathbb{E}[R(\tau)] = E\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, A_t\right]$$

- Policy Gradient는 현재 정책의 기댓값을 추정함
- 즉, 같은 trajectory를 반복 사용하거나 현재 정책과 다른 분포의 데이터를 쓰면 gradient가 부정확해져 정책이 과도하게 변할 수 있음

# Trust Region Policy Optimization

## Objective

- TRPO에서는 **ratio**를 사용해서 과거 정책이 생성된 데이터로부터 현재 정책의 reward 기댓값을 추정한다.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

### TRPO Objective

$$\max \mathbb{E}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \quad where\ \mathbb{E}_t[KL(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \leq \delta$$

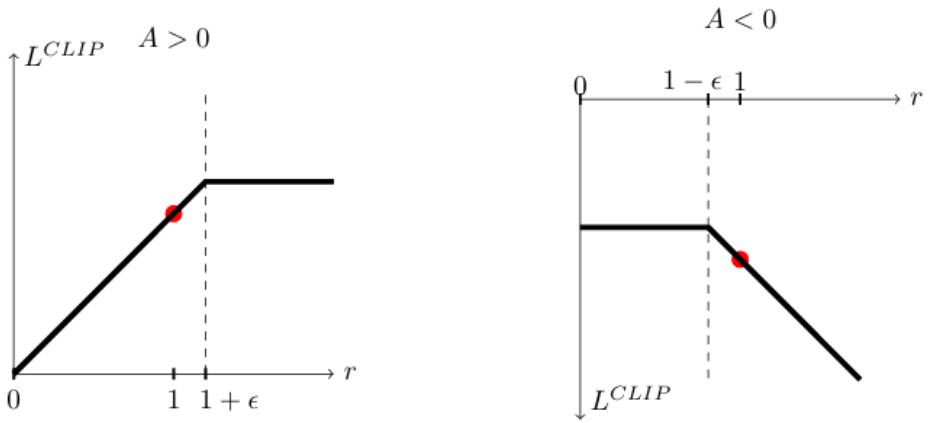
- KL divergence 제약 조건을 통해 정책의 극단적인 변화를 제한함
- 이 제약으로 인해 최적화 과정이 복잡해지고 연산 부담이 커짐
- 또한, dropout처럼 노이즈가 포함될 수 있는 모델 구조에는 적용이 어려움

# Proximal Policy Optimization

## Objective

### Clipped surrogate objective

$$\mathrm{L}^{\mathrm{CLIP}}(\theta) = \mathbb{E}_{\mathrm{t}}[\min(\mathrm{r}_{\mathrm{t}}(\theta)\mathrm{A}_{\mathrm{t}}, \mathrm{clip}(\mathrm{r}_{\mathrm{t}}(\theta), 1-\epsilon, 1+\epsilon)\mathrm{A}_{\mathrm{t}})]$$




- $\mathrm{A}_{\mathrm{t}} > 0$: $\mathrm{r}_{\mathrm{t}}(\theta)$ 이 너무 큰 경우 clipping 작동
- $\mathrm{A}_{\mathrm{t}} < 0$: $\mathrm{r}_{\mathrm{t}}(\theta)$ 이 너무 작은 경우 clipping 작동

# Proximal Policy Optimization

## Final Objective

- Combines three terms: clipped objective, value loss, and entropy bonus

$$L^{CLIP+VF+S}(\theta) = \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s)]$$

- **Clipped objective** improves training stability by constraining the update size

$$L^{CLIP}(\theta) = E_t[\min(x, clip(x, 1-\epsilon, 1+\epsilon)A_t]$$

- **Value loss** helps estimate how good each state is

$$L^{VF}(\theta) = (V_{target} - V_\theta(S))^2$$

- **Entropy bonus** encourages the policy to keep exploring

# Proximal Policy Optimization

## Summary

### Clipped Surrogate Objective

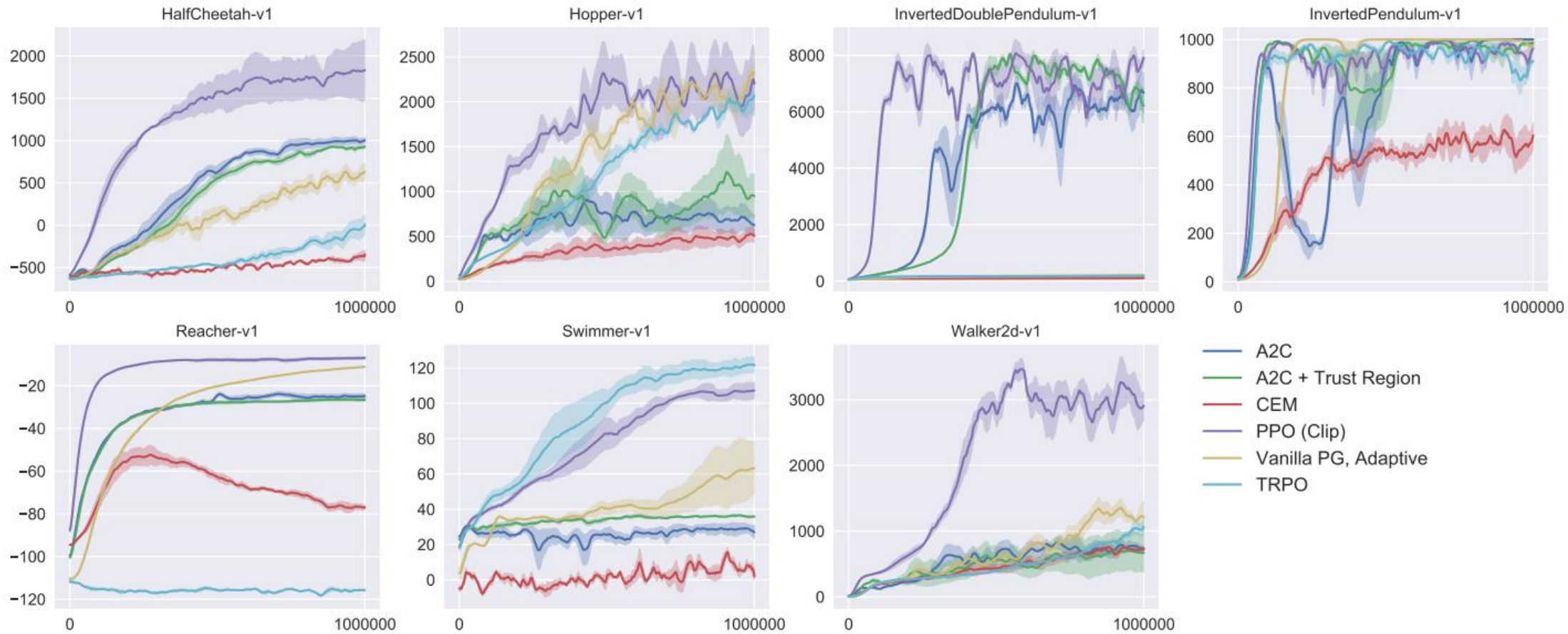
$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

### PPO Objective

$$L^{CLIP+VF+S}(\theta) = \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 s[\pi_\theta](s)]$$

# Experiments

## Comparison to Other Algorithms



# Conclusion

- PPO는 **정책이 급격하게 변하는 것을 막아** 안정적인 학습을 유도한다.

- TRPO의 아이디어를 바탕으로 하지만, **복잡한 KL 제약 최적화를 clipping 방식으로 단순화**해 구현이 훨씬 쉽다.

# Thank you

**Jeongwoo Kim**

(jeongwu510@gmail.com)


**질문** https://forms.gle/LtvyMJ7BFwMKpWtz8 **피드백** https://forms.gle/PAmxLQnRBZVhAMaw8

**응답** https://docs.google.com/spreadsheets/d/1uWyc0pUfQOwTTZUDyY3gxlmU5xcFro90kKJKOcDitnk/edit?usp=sharing

# Appendix: Actor-Critic Algorithm

- **From Advantage to Actor-Critic**
  - baseline $b(s_t)$를 직접 설계하는 대신 critic이 $V^{\pi}(s_t)$를 학습해서 추정함
  - actor는 이 advantage를 사용해 policy를 업데이트함
  - 따라서 actor와 critic을 함께 학습하는 구조가 됨

# Appendix: Actor-Critic Algorithm

## Key Terms

### Policy Gradient (Actor)

- The policy denoted as $\pi(a \mid s)$, represents the probability of taking action a in state s..

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=0}^{N} \nabla_\theta \log \pi_\theta (a_i|s_i) \cdot A(s_i,a_i)$$

- $J(\theta)$ represents the expected return under the policy parameterized by $\theta$
- $\pi_\theta(a \mid s)$ is the policy function
- $A(s, a)$ is the advantage function representing the advantage of taking action a in state s

# Appendix: Actor-Critic Algorithm

## Key Terms

### Value Function Update (Critic)

- The value function, denoted as $V(s)$ ,estimates the expected cumulative reward starting from state s.

$$\nabla_{\omega} J(\omega) \approx \frac{1}{N} \sum_{i=0}^{N} \nabla_{\omega} (V_{\omega}(s_i) - Q_{\omega}(s_i, a_i))^2$$

- $\nabla_{\omega} J(\omega)$ is the gradient of the loss function with respect to the critic's parameters $\omega$
- $V_{\omega}(s_i)$ is the critic's estimate of value of state s with parameter w
- $Q_{\omega}(s_i, a_i)$ is the critic's estimate of the action-value of taking action a

# Appendix: Actor-Critic Algorithm

## Update Rules

### Actor Update

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

- $\alpha$ is learning rate for the actor
- t is the time step within an episode

### Critic Update

$$\omega_t = \omega_t - \beta \nabla_\omega J(\omega_t)$$

- w represents the parameters of the critic network
- $\beta$ is the learning rate for the critic

# Actor-Critic Algorithm

- Update Rules