

# Asynchronous Methods for Deep Reinforcement Learning

Deep Learning Paper Review

한양대학교 AILAB 김 건

# Introduction



Introduction



Related Work



Model



Experiments



Conclusion

- [Asynchronous Methods for Deep Reinforcement Learning](#)
- ICML 2016에서 발표된 논문
- DQN을 개선한 논문
- Asynchronous(비동기) Methods를 4가지 강화학습 방법들에 적용시킴
- 결과적으로 Asynchronous Advantage Actor-Critic RL(A3C)을 제안
- 1 Machine에 Multi Thread로 구현
- A3C 알고리즘은 여러 환경에서 기존 기법들과 비교하여 더 적은 Computation Resource 로 (16cpu without gpu) 더 우수한 결과를 (여러 게임에서 outperform) 수십 배까지 빠른 속도로 학습함
- <https://www.youtube.com/watch?v=gINks-YCTBs> 좋은 수식 설명



Introduction



Related Work



Model



Experiments

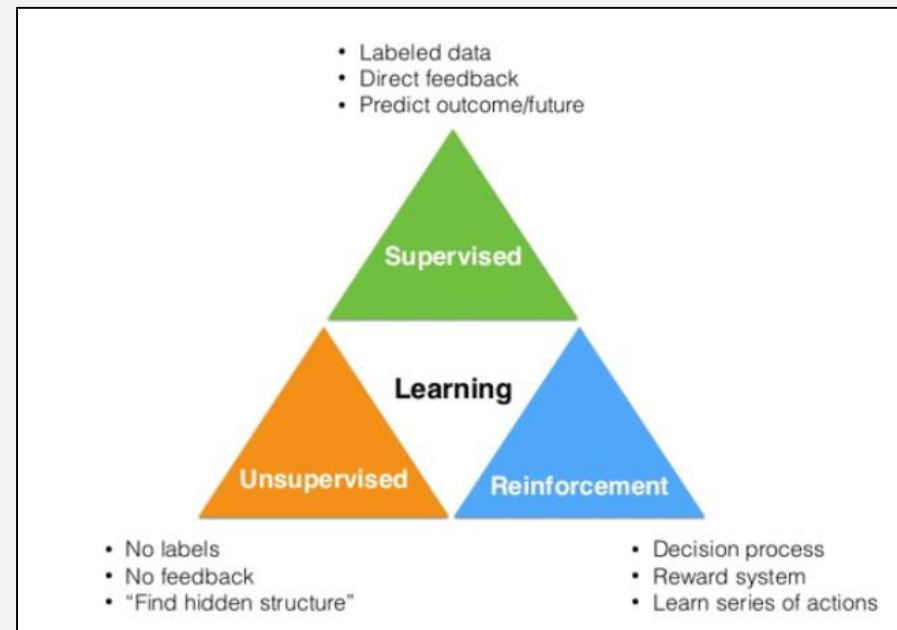
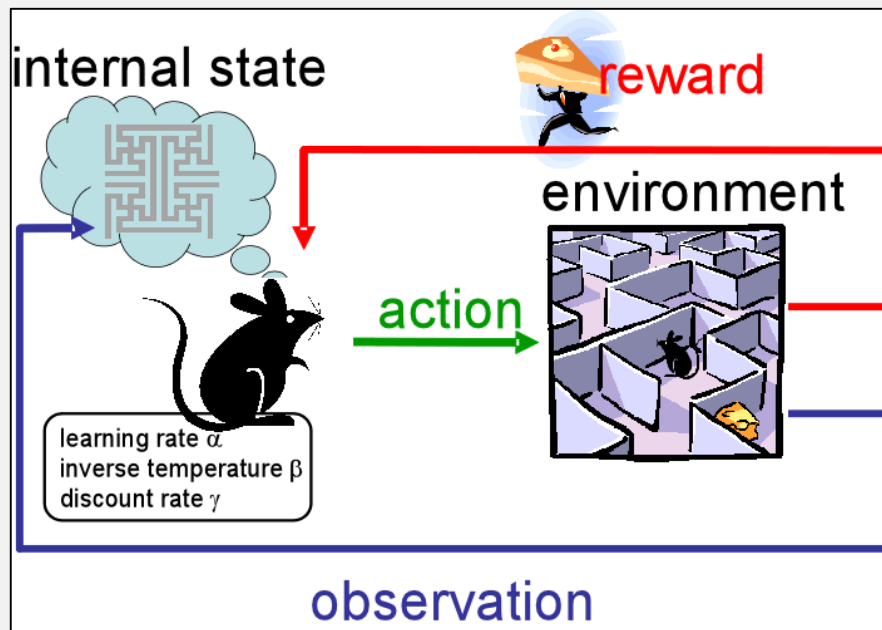


Conclusion

- Background
  - Reinforcement Learning
  - Policy & Value Function
  - Q-learning
- Related Work
  - DQN


## 강화 학습?

- 어떤 환경(**Environment**) 안에서 정의된 에이전트(**Actor**)가 현재의 상태(**State**)를 인식하여, 선택 가능한 행동들(**Actions**) 중 보상(**Reward**)을 최대화 하는 행동 혹은 행동순서를 선택(**Policy**)하는 방법이다. (Wikipedia)



- 정답은 모르지만, 자신이 한 행동에 대한 보상을 통해 학습

# Policy & Value Function

 Introduction Related Work Model Experiments Conclusion

- **Policy -  $\pi(s)$** 
  - 어떤 state에서 어떤 action을 할 것인가
  - Optimal Policy (보상을 최대화 하는 Policy) 를 찾는 것이 강화 학습의 목적
- **State-value function**
  - 어떤 Policy를 진행하였을 때 어떤 상태  $s$ 의 가치
- **Action-value function**
  - 어떤 상태  $s$ 에서 action  $a$ 를 할 때의 보상
- 위의 value function들을 이용하여 policy를 improve한다.

## Definition

The **state-value function  $v_\pi(s)$**  of an MDP is the expected return starting from state  $s$ , and then following policy  $\pi$

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

## Definition

The **action-value function  $q_\pi(s, a)$**  is the expected return starting from state  $s$ , taking action  $a$ , and then following policy  $\pi$

$$q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a]$$



Introduction



Related Work



Model



Experiments



Conclusion

## Q-function(action value function)



$$Q(\text{state}, \text{action}) = r$$

state와 action을 넣어주면 reward 값을 주는 함수

Exploration , Discounted future reward  
두 개념을 적용하여 Q function을 학습

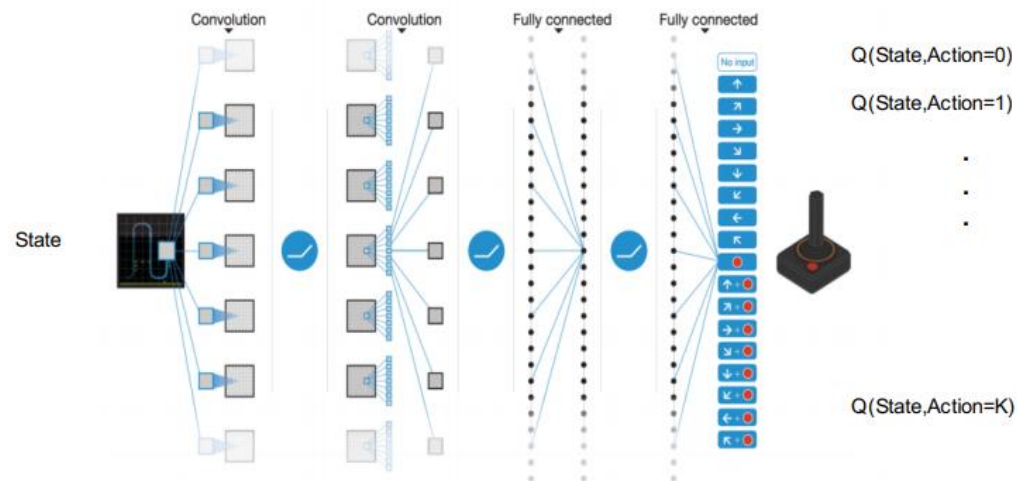
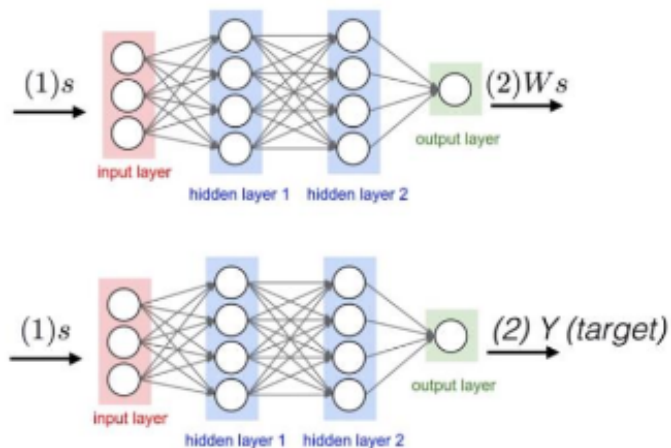
```
Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):  
  Initialize  $S$   
  Repeat (for each step of episode):  
    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
    Take action  $A$ , observe  $R, S'$   
     $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$   
     $S \leftarrow S'$ ;  
  until  $S$  is terminal
```

- DQN ([2013](#), [2015](#)) Deep Q Network

- CNN을 활용하여 pixel data 자체를 학습 가능 (한 agent 여러 게임 적용)
- Experience replay memory** 를 활용하여 input들 간의 correlation으로 인해 발생하는 문제를 해결 (학습의 비효율성, policy 고정)
- Target network**를 분리하여 target의 불안정성을 해결함

$$\min_{\theta} \sum_{t=0}^T [\hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta))]^2$$

$$\min_{\theta} \sum_{t=0}^T [\hat{Q}(s_t, a_t | \theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta))]^2$$







Introduction



Related Work



Model



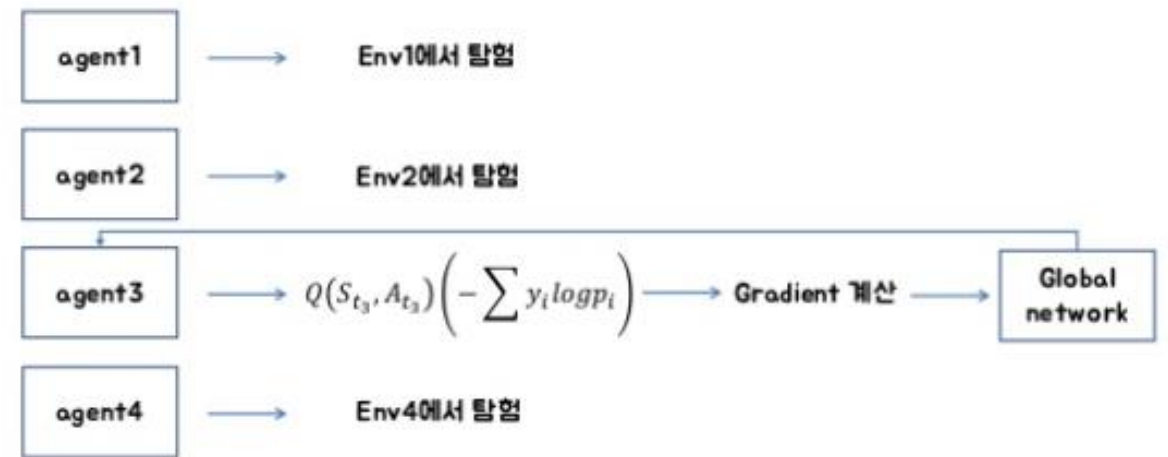
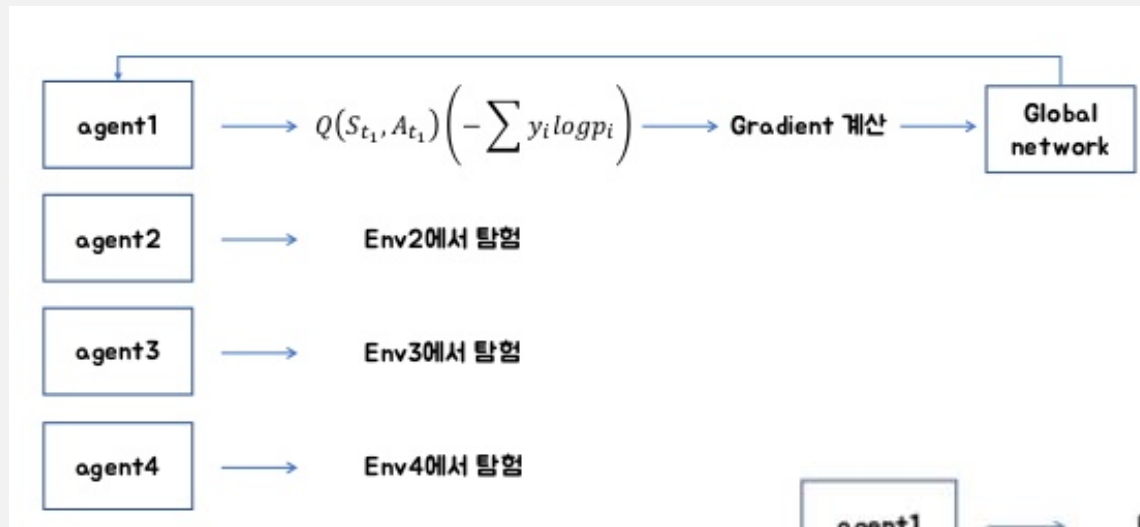
Experiments



Conclusion

- DQN의 단점
  1. 많은 메모리의 사용
  2. 느린 학습 속도
  3. 불안정한 학습 과정 (value-based greedy policy)
- **Asynchronous Advantage Actor-Critic (A3C)** RL 으로 모든 문제를 해결함

- 여러 개의 에이전트를 만들어서 각각 gradient 를 계산
- 비동기적으로 global network를 업데이트 한다.



# Asynchronous Methods

 Introduction

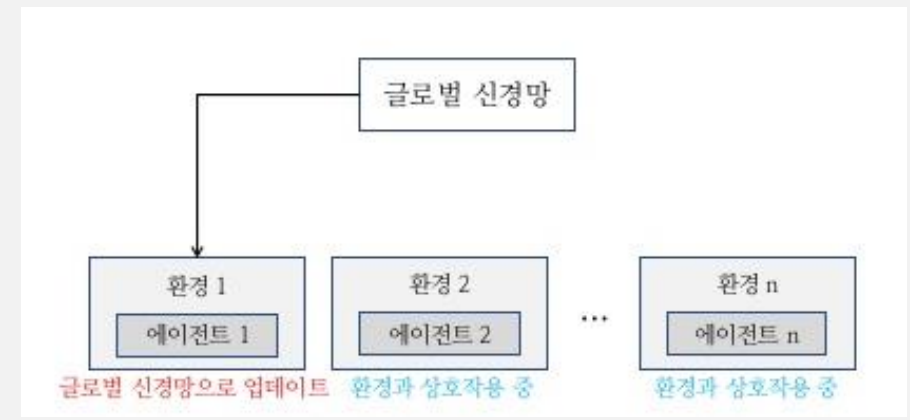
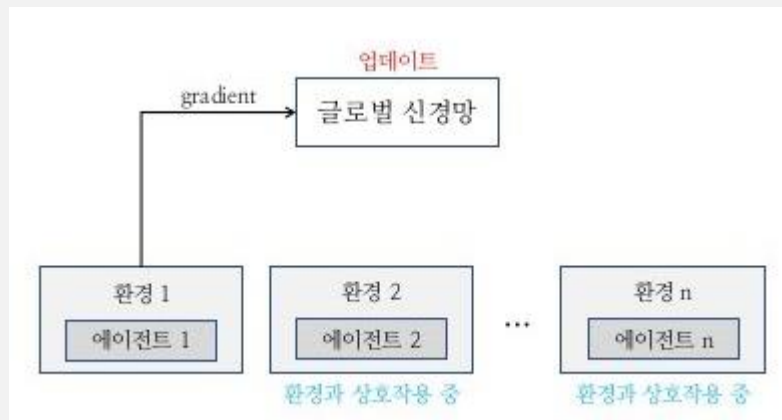
 Related Work

 Model

 Experiments

 Conclusion

- 각 agent가 다른 policy를 가지고 탐험을 하기에 replay memory를 사용할 필요가 없어짐 (각 agent가 서로 다른 exploration policy 가짐)
- Monte Carlo -> TD 사용 가능
- Monte Carlo Method
  - 한 Episode 마다 학습을 진행하는 방법
- Temporal Difference
  - Time step 마다 학습을 진행하는 방법



# Actor - Critic

 Introduction

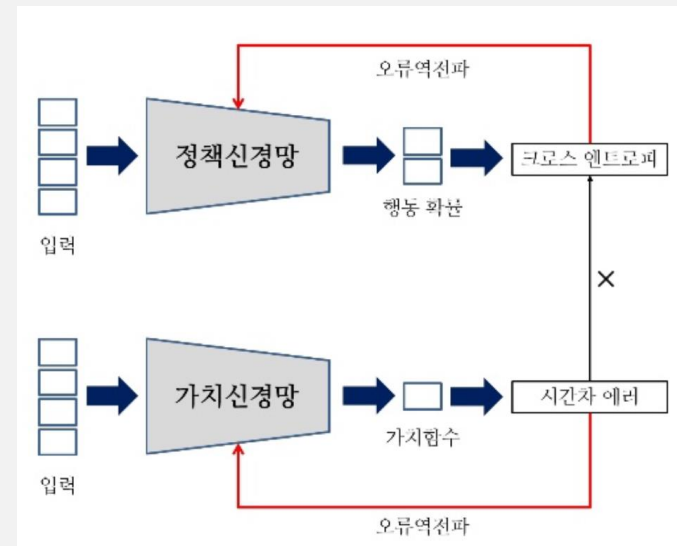
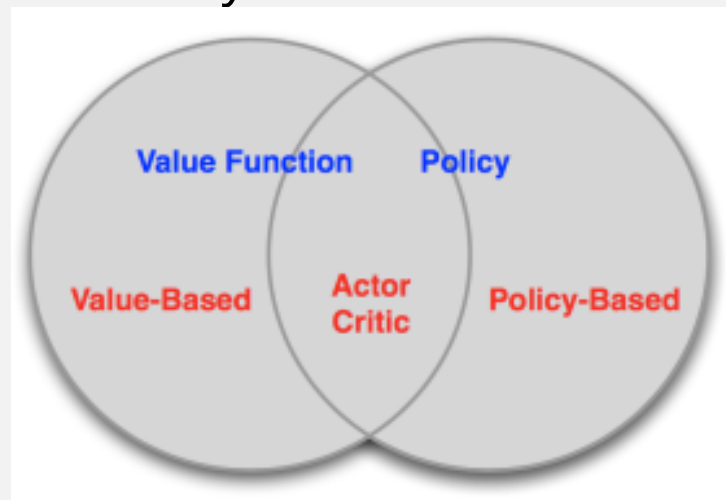
 Related Work

 Model

 Experiments

 Conclusion

- Actor-Critic = 실시간 학습 + Policy Gradient
- Policy-Based RL
  - Policy 자체를 학습하는 방식
  - 경로 동안 받을 것이라고 기대하는 보상의 합
  - 기존 Value-Based RL은 Value function을 학습하고 거기에 Policy를 맞췄다.

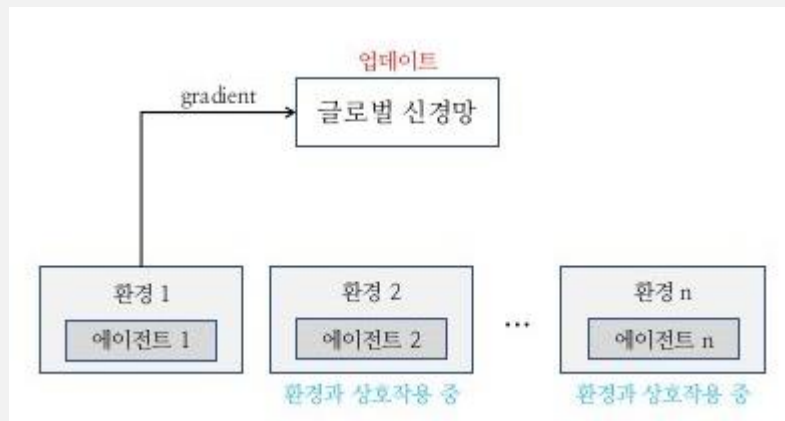


# Actor – Critic & Advantage

 Introduction Related Work Model Experiments Conclusion

- Critic
  - Value function을 근사하여 현재 상태를 Evaluation
- Actor (agent) 행동을 함
  - Critic이 제안하는 방향으로 정책을 근사
    - **Policy** -  $\pi(s)$ 
      - 어떤 state에서 어떤 action을 할 것인가
      - Optimal Policy (보상을 최대화 하는 Policy) 를 찾는 것이 강화 학습의 목적
    - **State-value function**
      - 어떤 Policy를 진행하였을 때 어떤 상태  $s$ 의 가치
- Advantage function
  - Actor–Critic 기법에 Advantage가 포함되 있다고 생각하면 됨
  - 예상치보다 실제로 더 나온 값. Loss에서 사용
  - Critic 으로 Actor를 평가한다.

- Actor-Critic Network를 여러 Thread에 생성하여 비동기적으로 학습을 함.
- DQN의 단점
  1. 많은 메모리의 사용
  2. 느린 학습 속도
  3. 불안정한 학습 과정 (value-based greedy policy)
- 1,2번 문제를 Asynchronous Methods를 통하여 해결
- 3번 문제를 Actor-Critic 기법을 적용하여 해결



Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

Table 1. Mean and median human-normalized scores on 57 Atari games using the human starts evaluation metric. Supplementary Table SS3 shows the raw scores for all games.

Method ↕	Number of threads ↕				
	1 ↕	2 ↕	4 ↕	8 ↕	16 ↕
1-step Q ↕	1.0 ↕	<b>3.0</b> ↕	<b>6.3</b> ↕	<b>13.3</b> ↕	<b>24.1</b> ↕
1-step SARSA ↕	1.0 ↕	<b>2.8</b> ↕	<b>5.9</b> ↕	<b>13.1</b> ↕	<b>22.1</b> ↕
n-step Q ↕	1.0 ↕	<b>2.7</b> ↕	<b>5.9</b> ↕	<b>10.7</b> ↕	<b>17.2</b> ↕
A3C ↕	1.0 ↕	2.1 ↕	3.7 ↕	6.9 ↕	12.5 ↕

일정 점수에 도달하기 위한 학습 속도  
1-step 방법이 더 많은 actor를 사용할 때  
특정 점수를 얻기 위해 더 적은 데이터가  
필요함을 관찰함.

1-step method의 bias를 줄이는데 multi-  
thread 방식이 아주 효과적이라서 그런 거  
같다고 함

- 당연한 얘기지만, 다른 모델보다 성능이 잘 나온다고 함
- [TORCS Car Racing Simulator](#)
- [MuJoCo Physics Simulator](#)
- [Labyrinth](#)



Introduction



Related Work



Model



Experiments



Conclusion

- Conclusion

- 기존 4개의 대표적인 강화 학습 알고리즘의 비동기 버전을 제시
- Value-based, policy-based, on & off policy methods 들을 사용한 다양한 도메인에서 안정적이고 효율적인 방식으로 학습이 가능함
- Experience replay도 통합하면 데이터 효율성을 향상시킬 수 있음
- 여러 강화 학습 기법들을 통합할 수 있을 거다.





Thank you