Policy Gradient & Actor-Critic

Slides from

1. 이웅원 외, 파이썬과 케라스로 배우는 강화학습, 주교재
2. 이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT
3. David Silver, Reinforcement Learning, PPT

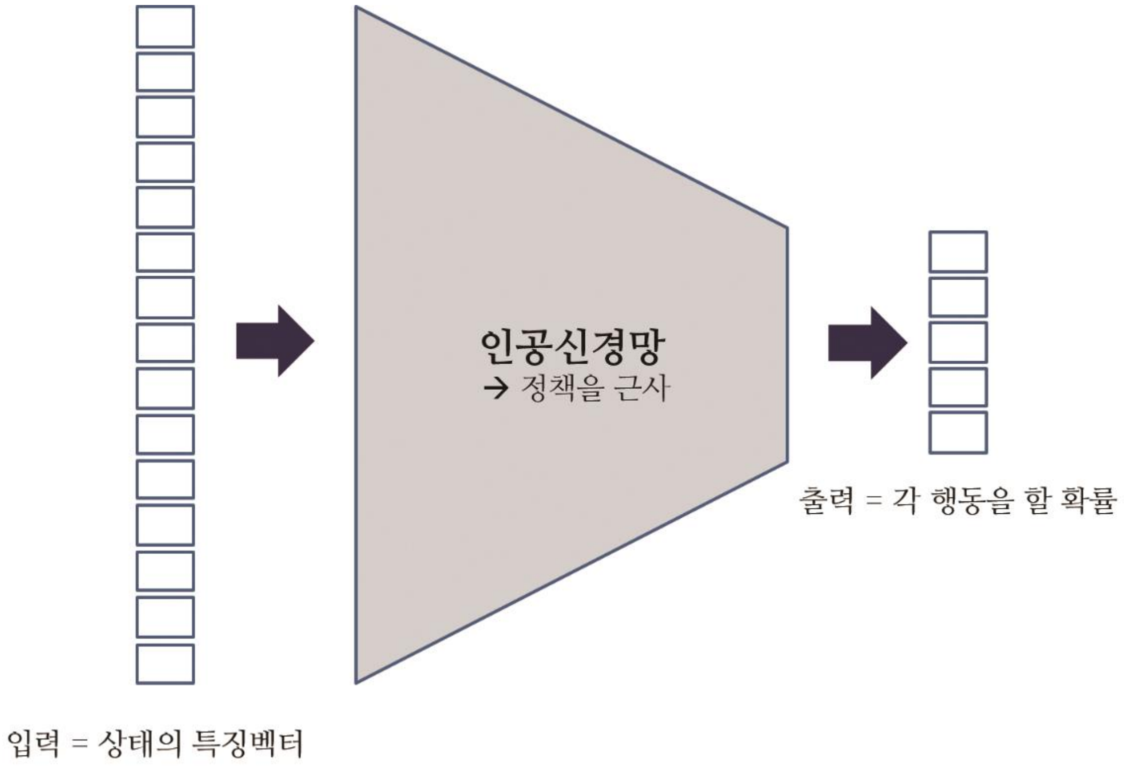
References

1. Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press
2. 유튜브, 전민영, 노승은, 강화학습의 기초 이론, 팡요랩

Contents

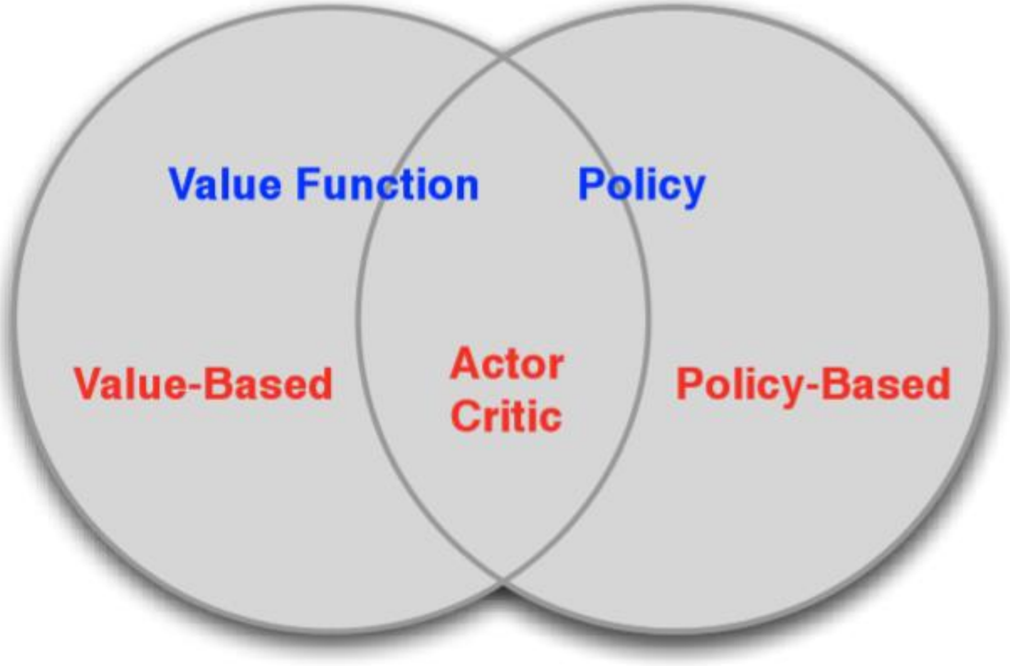
정책 기반 강화학습

- 지금까지의 강화학습 : 가치 기반 (Value-based)
  - 에이전트가 가치함수를 기반으로 행동을 선택하고 가치함수를 업데이트하면서 학습
  - 인공신경망이 큐함수를 근사 -> 가치신경망
  - 출력층의 활성함수가 선형함수
- 이번 강의의 강화학습 : 정책 기반 (Policy-based)
  - 가치함수를 토대로 행동을 선택하지 않고 정책을 직접적으로 근사시킴
  - 인공신경망이 정책을 근사 -> 정책신경망
  - 출력층의 활성함수로 Softmax 함수 사용



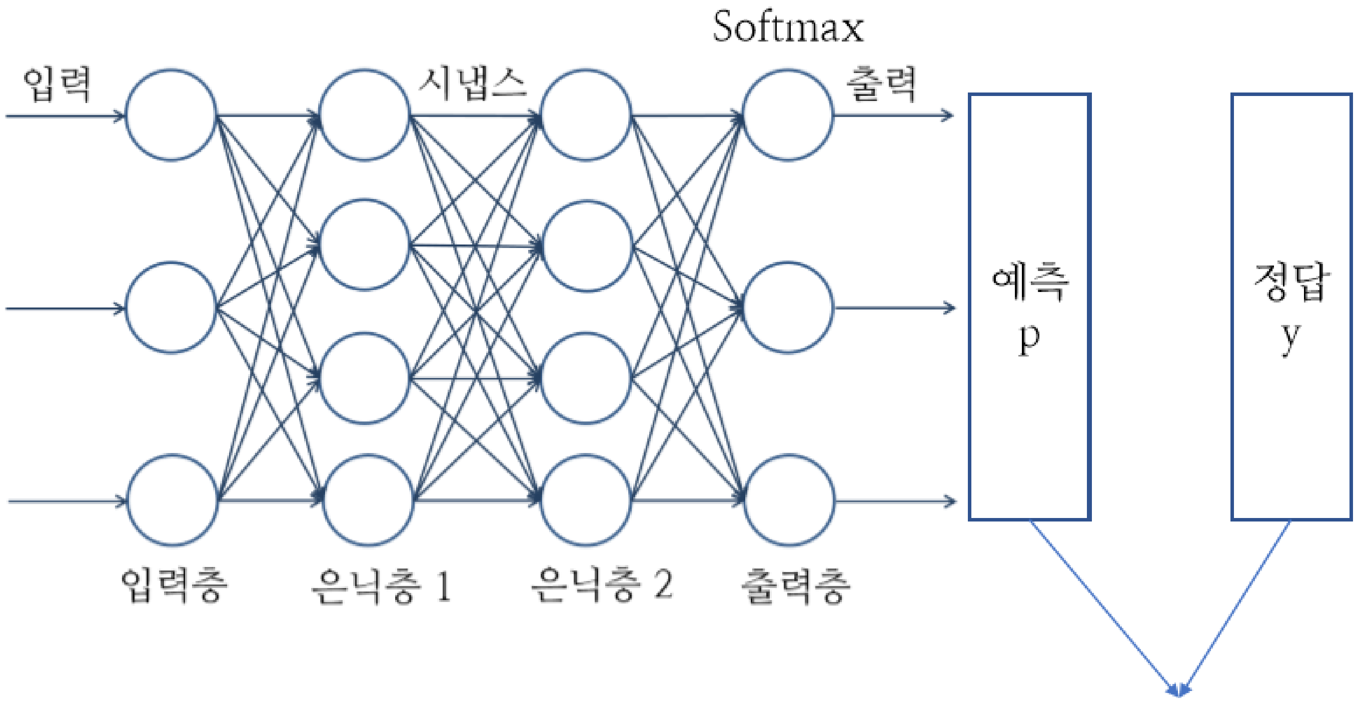
Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



David Silver, Reinforcement Learning, PPT

신경망 & 지도학습 & 크로스 엔트로피



$-\sum y_i log p_i$: **p와 y가 얼마나 다른지**

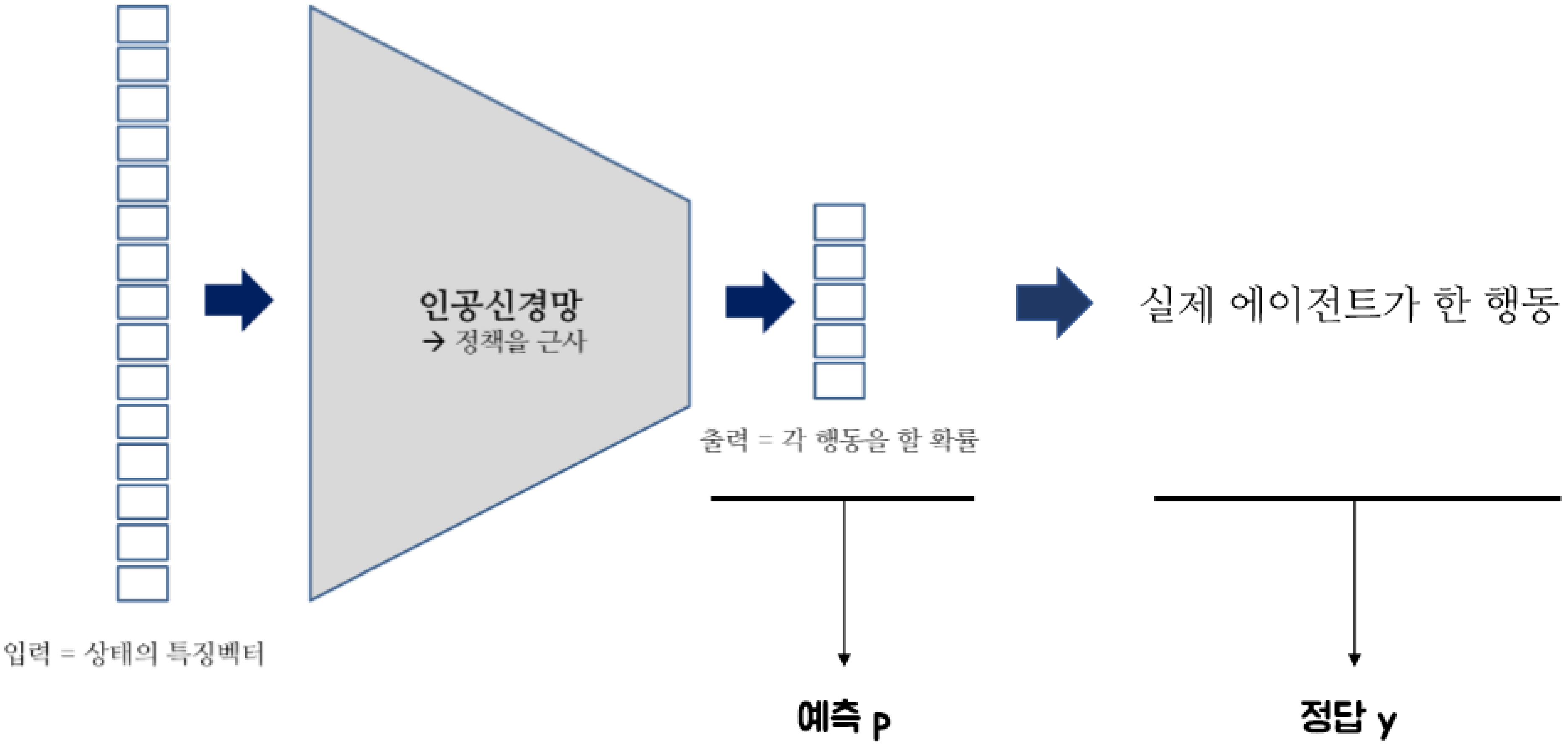
- Regression
  - Continuous output -> 큐함수
  - Linear activation function
  - MSE (Mean Square Error)
- Classification
  - Discrete Output -> 정책
  - Softmax activation function & One hot encoding
  - CEE (Cross Entropy Error)

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

신경망 & 지도학습 & 크로스 엔트로피

Input data | Label $\rightarrow$ $-\sum y_i log p_i$ 계산 $\rightarrow$ Gradient descent

강화학습 & 크로스 엔트로피



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

강화학습 & 크로스 엔트로피

상태
$s_0, s_1, s_2, \cdots$

실제 행동
$a_0, a_1, a_2, \cdots$

$\longrightarrow$ $-\sum y_i log p_i$ **계산** $\longrightarrow$ **Gradient descent**

나의 예측을 실제 행동에 가깝게 만든다 → 어떤 의미??

업데이트의 방향성이 필요!

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

강화학습 & 크로스 엔트로피

정답이 되는 각 행동이 실제로 좋은 지, 좋다면 얼마나 좋은 지를 알 수 있을까?

→ **큐함수(Q-function)** : $Q_\pi(s,a) = \boldsymbol{E}_\pi[R_{t+1} + \gamma R_{t+2} + \cdots | S_t = s, A_t = a]$

상태 **(S_t)**

실제 행동 **(A_t)**

→ $Q(S_t, A_t)(-\sum y_i log p_i)$**계산** → **Gradient descent**

업데이트의 방향과 크기

크리틱 : 뉴럴넷

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

Policy Gradient

- 정책의 표현

  policy = $\pi_{\theta}(\mathrm{a}|s) = \mathbf{P}[\mathrm{a}|\mathrm{s}, \theta]$, θ: 정책신경망의 가중치

- 목표함수 – 누적보상

$$J(\theta) = \boldsymbol{E}[R_1 + \gamma R_2 + \cdots + \gamma^{T-1} R_T \mid \pi_{\theta}]$$

- 정책기반 강화학습의 목표 – maximize $J(\theta)$

  --> 경사상승법 (Gradient Ascent)

- Gradient Ascent

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

Policy Objective Functions

- In episodic environments we can use the start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- In continuing environments we can use the average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or the average reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathcal{R}_s^a$$

- where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

David Silver, Reinforcement Learning, PPT

Policy Gradient

- 목표함수를 가치함수로 나타내면
  1) 에피소드의 시작 상태 s_0 를 기준으로 나타내면

$$J(\theta) = v_{\pi_\theta}(s_0)$$

  2) 모든 상태에 대해 평균 값을 기준으로 나타내면

$$\boldsymbol{J(\theta) = \sum_s d_{\pi_\theta}(s)\, v_{\pi_\theta}(s),}$$

where $d_{\pi_\theta}(s)$: 에이전트가 상태 s 에 있을 확률 (상태 분포)

- 목표함수의 미분 : Policy Gradient

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_s d_{\pi_\theta}(s)\, v_{\pi_\theta}(s)$$

수식 전개

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_s d_{\pi_\theta}(s)\, v_{\pi_\theta}(s)$$

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(\mathrm{a|s}) q_\pi(s,a)$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(\mathrm{a|s}) \times \frac{\nabla_\theta \pi_\theta(\mathrm{a|s})}{\pi_\theta(\mathrm{a|s})}\, q_\pi(s,a)$$

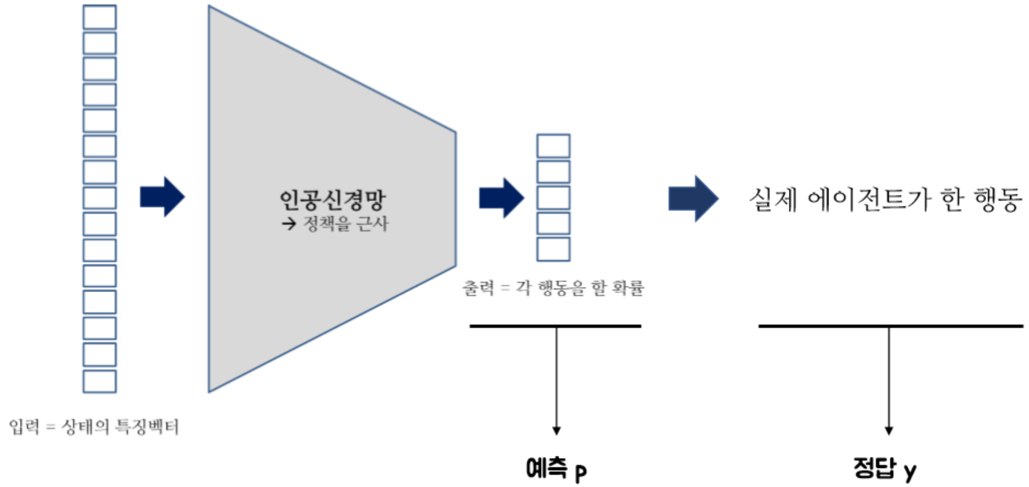
$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(\mathrm{a|s}) \times \nabla_\theta \log \pi_\theta(\mathrm{a|s})\, q_\pi(s,a)$$

$$\boldsymbol{\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta\, log\, \pi_\theta(a|s) q_\pi(s,a)]}$$

-> 기대값은 샘플링으로 대체할 수 있음

Policy Gradient & Cross Entropy



- Cross Entropy

$$-\sum y_i \log p_i \rightarrow -\log \pi_\theta(\mathrm{a}|\mathrm{s})$$

p_i : 각 행동을 할 확률 -> $\pi_\theta(\mathrm{a}|\mathrm{s})$
y_i : 실제 에이전트가 한 행동 -> 모든 a 중에 하나만 1이고 나머지는 0

- Policy Gradient

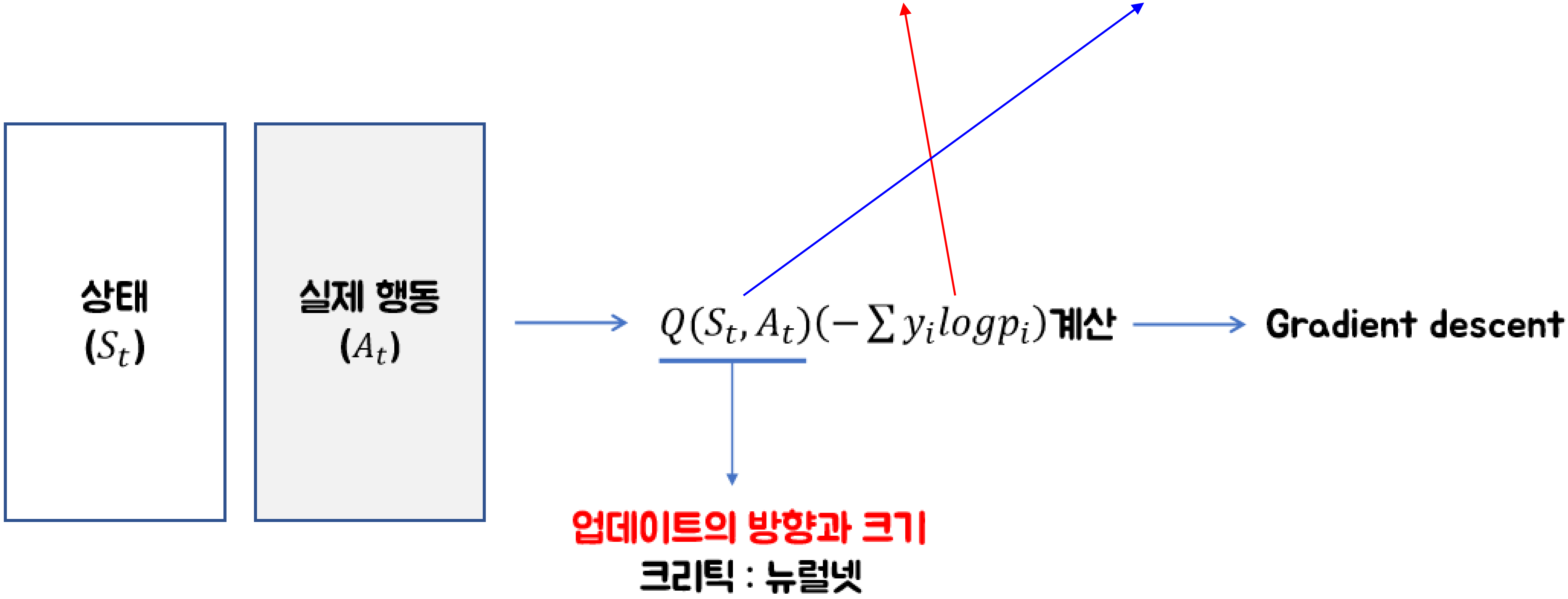
$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\mathrm{a}|\mathrm{s}) q_\pi(s,a)] = E_{\pi_\theta}[-\nabla_\theta \textcolor{red}{-\log \pi_\theta(\mathrm{a}|\mathrm{s})} \textcolor{blue}{q_\pi(s,a)}]$$

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

Policy Gradient & Cross Entropy

- Policy Gradient

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[-\nabla_\theta \, -\log \pi_\theta(\mathrm{a}|\mathrm{s}) q_\pi(s,a)]$$



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

Policy Gradient Ascent

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\mathrm{a|s}) q_\pi(s,a)]$$

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

--> 문제점 :

에이전트는 정책만 가지고 있고 가치함수 혹은 큐함수는 가지고 있지 않음

REINFORCE 알고리즘

- 큐함수를 반환값 G_t 로 대체

$q_\pi(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}[G_t|S_t = s, A_t = a]$ (2-2 Bellman Equation.PPT)

- 경사상승법 -> 경사하강법

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

- 에피소드 마다 실제로 얻은 보상으로 학습 --> Monte Carlo Policy Gradient

REINFORCE 알고리즘

1. 한 에피소드를 현재 정책에 따라 실행
2. Trajectory를 기록

$$\tau = s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_T$$

3. 에피소드가 끝난 뒤 G_t를 계산
4. Policy Gradient를 계산해서 정책 업데이트
5. (1~4) 반복

REINFORCE 알고리즘

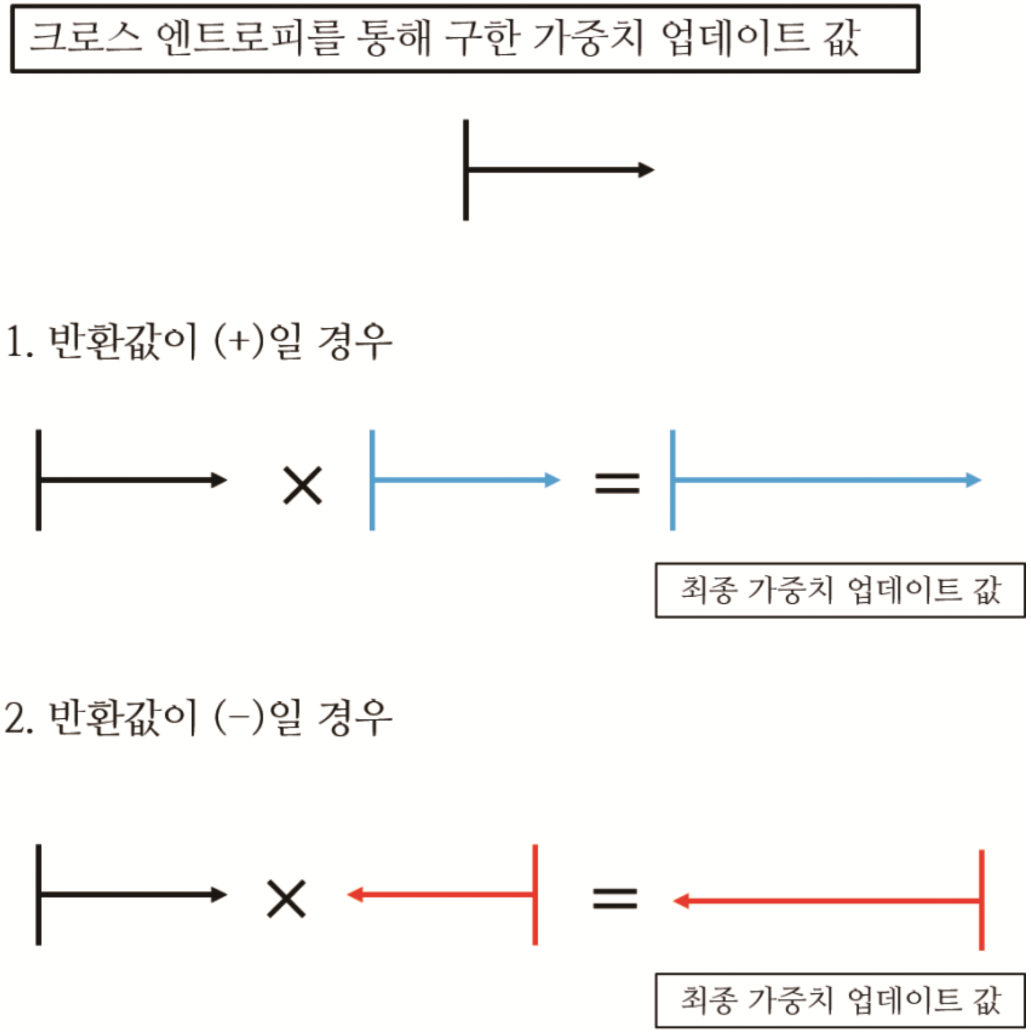
- REINFORCE 알고리즘에서 정책 신경망의 가중치 업데이트 값을 구하는 과정

➢ 크로스엔트로피 : $-\log \pi_\theta(a|s)$

➢ 반환값 : G_t

➢ Policy Gradient :

$$\nabla_\theta -\log \pi_\theta(a|s)\, G_t$$



REINFORCE 알고리즘

REINFORCE의 문제

1. Variance가 높다
2. 에피소드마다 업데이트 (on-line X)

몬테카를로 --> TD (Temporal-Difference)

REINFORCE --> Actor-Critic

Actor-Critic

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) q_\pi(s,a)]$$

- We use a critic to estimate the action-value function,

$$Q_w(s,a) \approx q_\pi(s,a)$$

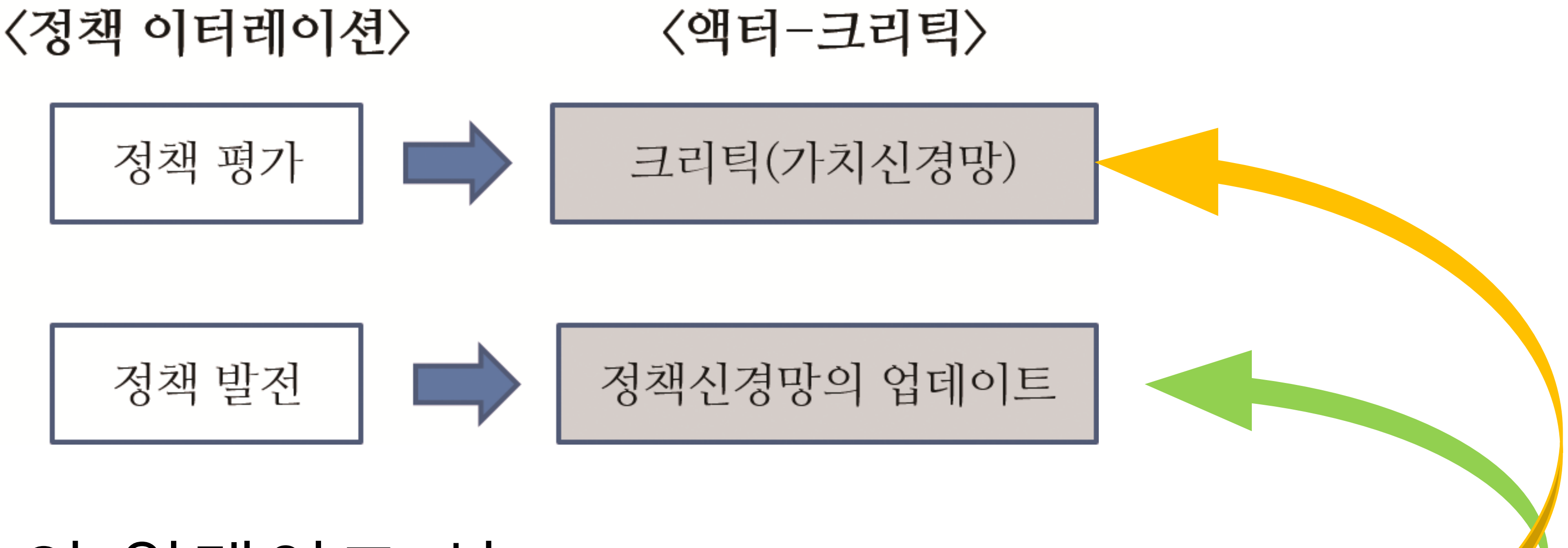
- Actor-critic algorithms maintain *two* sets of parameters
  - Critic Updates action-value function parameters w
  - Actor Updates policy parameters θ, in direction suggested by critic
- Actor-critic algorithms follow an *approximate* policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\; Q_w(s,a)]$$

David Silver, Reinforcement Learning, PPT

Actor-Critic

- 에피소드마다가 아닌 타임스텝마다 학습
- 다이나믹 프로그래밍의 정책 이터레이션 구조를 사용



- Policy Gradient의 업데이트 식

$$\theta_{t+1} \approx \theta_t - \alpha[-\nabla_\theta \log \pi_\theta(\mathrm{a}|\mathrm{s}) Q_W(s,a)]$$

Actor-Critic

- Actor-Critic 업데이트 식

$$\theta_{t+1} \approx \theta_t - \alpha[-\nabla_\theta \log \pi_\theta(\mathrm{a}|\mathrm{s}) Q_W(s,a)]$$

- Actor-Critic 오류함수

오류함수 = 정책신경망 출력의 크로스 엔트로피 × 큐함수(가치신경망 출력)

$$= -\log \pi_\theta(\mathrm{a}|\mathrm{s}) \times Q_W(s,a)$$

- Policy Gradient : $\nabla_\theta - \log \pi_\theta(\mathrm{a}|\mathrm{s}) Q_W(s,a)$

-> 큐함수의 값에 따라 오류함수의 값이 많이 변화 -> 분산이 큼

-> 베이스라인 사용

Reducing Variance Using a Baseline

- We subtract a baseline function $B(s)$ from the policy gradient
- This can reduce variance, without changing expectation

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] = \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a) B(s)$$

$$= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s) \nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a)$$

$$= 0$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_\pi(s,a)$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) \times \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} q_\pi(s,a)$$

$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}$$

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) \times \nabla_\theta \log \pi_\theta(a|s)\, q_\pi(s,a)$$

$$\boldsymbol{\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) q_\pi(s,a)]}$$

- A good baseline is the state value function $B(s) = V^{\pi_\theta}(s)$
- So we can rewrite the policy gradient using the advantage function $A^{\pi_\theta}(s,a)$

$$\boldsymbol{\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_W(s,a)]}$$

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^{\pi_\theta}(s,a)\right]$$

David Silver, Reinforcement Learning, PPT

Actor-Critic

- Advantage 함수의 베이스라인으로 가치함수 V_v 사용

$$A(S_t, A_t) = Q_W(S_t, A_t) - V_v(S_t)$$

  -> 큐함수 Q_W 와 가치함수 V_v 를 따로 근사하므로 비효율적

  -> Critic에 대해 Q_W 와 V_v 의 각각 다른 신경망 구축 필요

- 큐함수는 가치함수로 표현 가능 (2-2 Bellman Equation 참조)

$$q_\pi(s,a) = R_s^a + \gamma v_\pi(s')$$

- 큐함수를 가치함수로 표현하면 시간차 에러가 됨 -> 1개의 신경망으로 통합

$$\delta_v = R_{t+1} + \gamma V_v(S_{t+1}) - V_v(S_t)$$

- Advantage 함수를 사용한 Actor-Critic의 업데이트 식

$$\boldsymbol{\theta_{t+1} \approx \theta_t + \alpha[\nabla_\theta \log \pi_\theta(a|s)\, \delta_v]}$$

Advantage Actor-Critic (A2C)

- 정책신경망 학습

  오류함수 = 크로스 엔트로피 × 시간차 에러

  $= -\log \pi_\theta(\mathrm{a|s}) \times \delta_v$

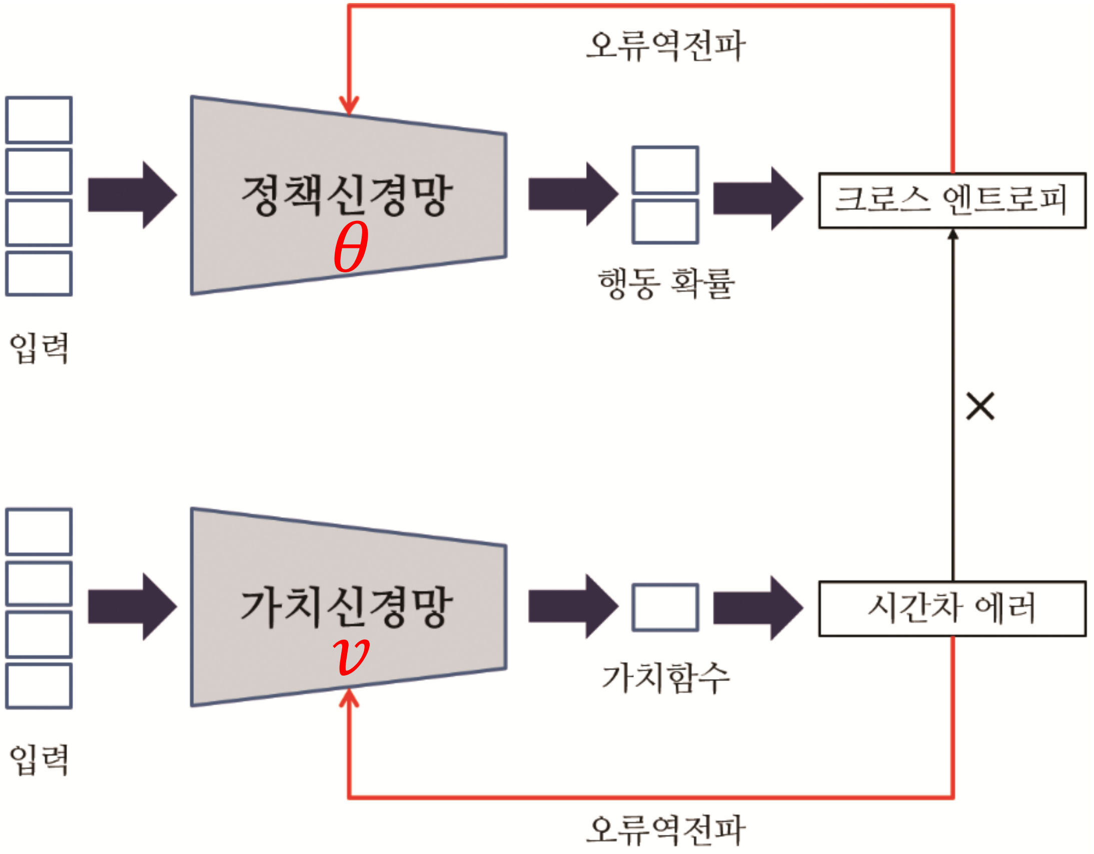
  Policy Gradient $= \nabla_\theta - \log \pi_\theta(\mathrm{a|s}) \times \delta_v$

- 가치신경망 학습 – 시간차 에러

  MSE = (정답 – 예측)^2 = δ_v^2

  $= (R_{t+1} + \gamma V_v(S_{t+1}) - V_v(S_t))^2$

--> A2C (Advantage Actor-Critic)



Advantage Actor-Critic (A2C)

1. Actor

1) 정책을 근사: θ

2) $\nabla_\theta log\pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\ )$로 업데이트

2. Critic

1) 가치함수(Value function)을 근사: v

2) $\left(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\right)^2$의 오차함수로 업데이트

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

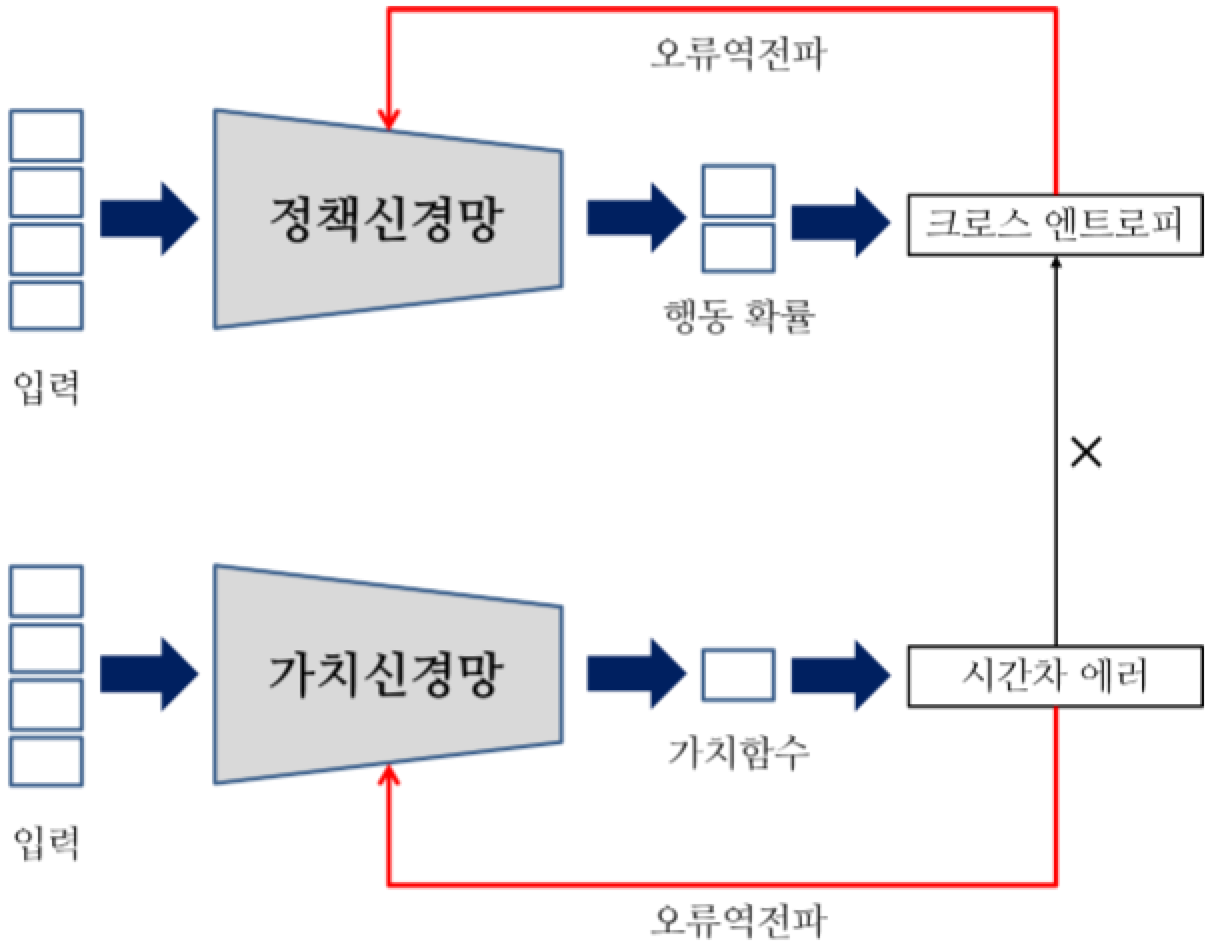
Advantage Actor-Critic (A2C)

1. Actor의 loss function

$$\underbrace{-\nabla_\theta log\pi_\theta(a_t|s_t)}_{\text{크로스 엔트로피}}\underbrace{(r_{t+1}+\gamma V_v(s_{t+1})-V_v(s_t)\ )}_{\text{시간차 에러}}$$

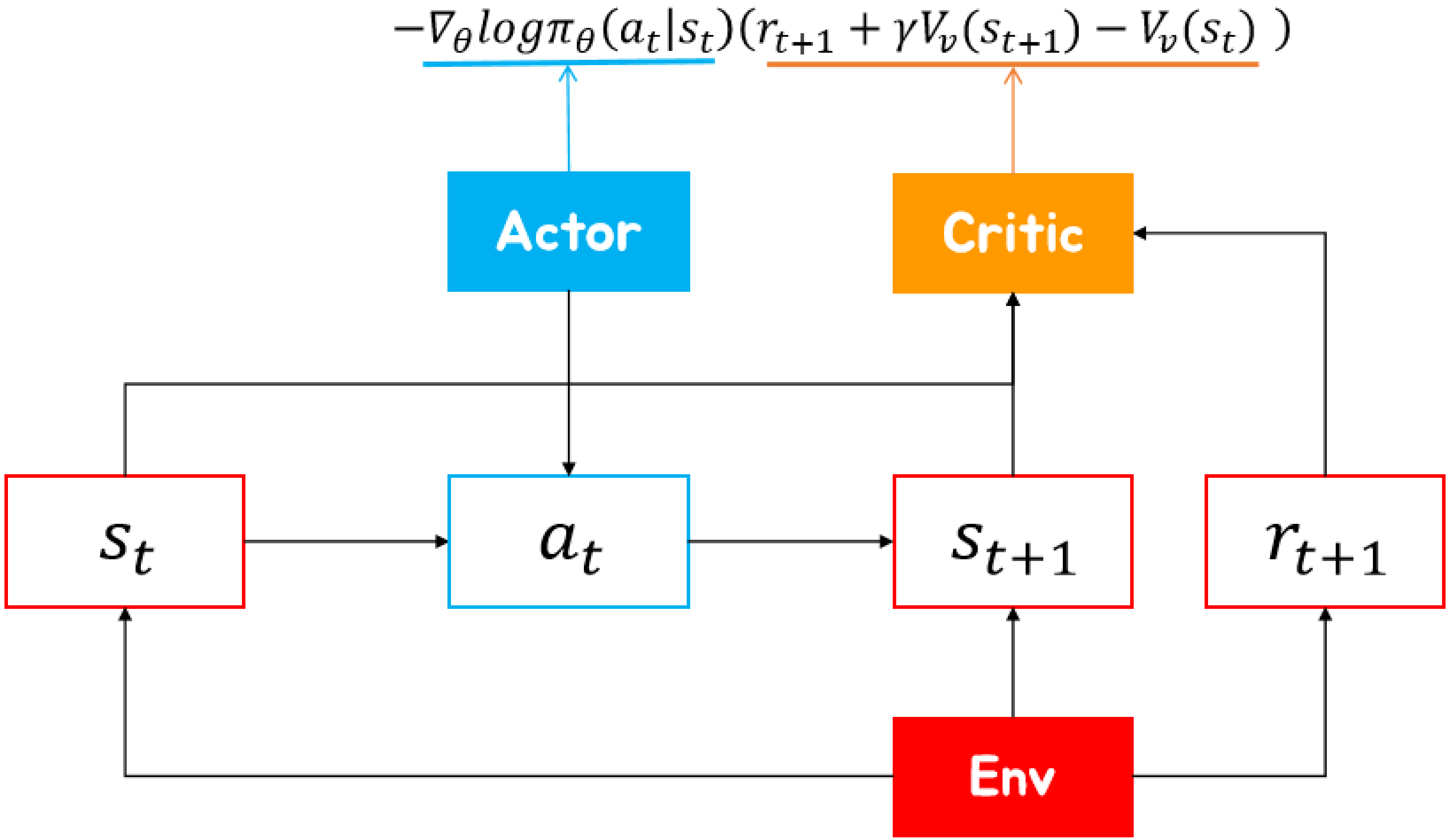
2. Critic의 loss function

$$\underbrace{\left(r_{t+1}+\gamma V_v(s_{t+1})-V_v(s_t)\right)^2}_{\text{시간차 에러}}$$



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

Advantage Actor-Critic (A2C)



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

Summary

- Policy Gradient
  - 인공신경망으로 정책을 근사하고 목표함수의 기울기를 따라 정책신경망을 업데이트
- REINFORCE 알고리즘 – MC 기반
  - Policy Gradient에서 반환값(G_t)을 이용해 에피소드마다 학습
  - Monte Carlo Policy Gradient
- Actor-Critic – TD 기반
  - 반환값(G_t) 대신에 큐함수를 인공신경망으로 근사하여 매 타임스텝마다 학습
  - Actor – 행동을 선택하는 인공신경망

$$\theta_{t+1} \approx \theta_t + \alpha[\nabla_\theta \log \pi_\theta(\mathrm{a|s})\, \delta_v]$$

  - Critic – 큐함수를 근사하여 각 행동이 얼마나 좋은지를 판단하는 인공신경망

$$\text{MSE} = (\text{정답} - \text{예측})^2 = (R_{t+1} + \gamma V_v(S_{t+1}) - V_v(S_t))^2$$

A3C = Asynchronous Advantage Actor-Critic

1. 샘플 사이의 상관관계를 비동기 업데이트로 해결
2. 리플레이 메모리를 사용하지 않음
3. policy gradient 알고리즘 사용가능 (Actor-Critic)
4. 상대적으로 빠른 학습 속도(여러 에이전트가 환경과 상호작용)

⇒ A3C = 비동기 + Actor-Critic

A3C

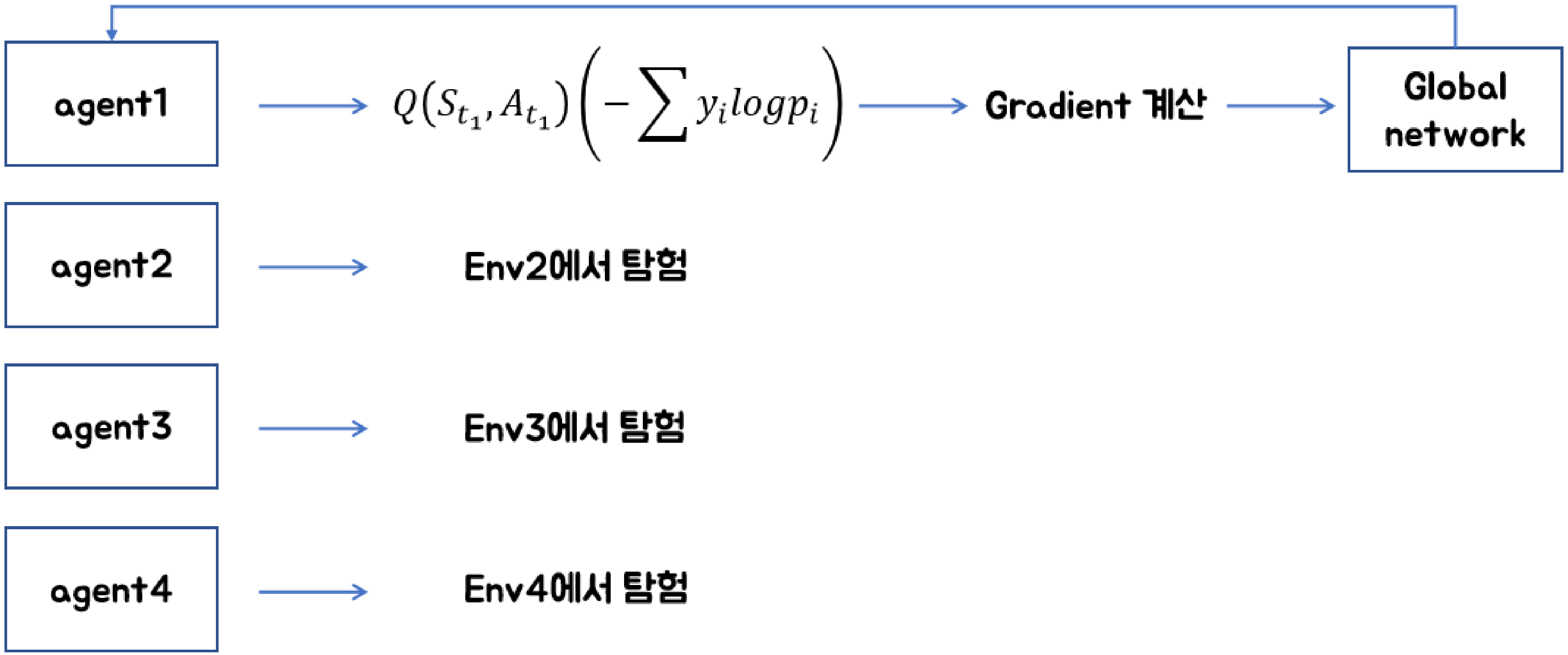
- 여러 개의 에이전트를 만들어서 각각 gradient를 계산하면?

agent1	Env1: 상태 (S_{t_1})	실제 행동 (A_{t_1})	$\longrightarrow Q(S_{t_1}, A_{t_1})\left(-\sum y_i log p_i\right) \longrightarrow$	Gradient 계산
agent2	Env2: 상태 (S_{t_2})	실제 행동 (A_{t_2})	$\longrightarrow Q(S_{t_2}, A_{t_2})\left(-\sum y_i log p_i\right) \longrightarrow$	Gradient 계산
agent3	Env3: 상태 (S_{t_3})	실제 행동 (A_{t_3})	$\longrightarrow Q(S_{t_3}, A_{t_3})\left(-\sum y_i log p_i\right) \longrightarrow$	Gradient 계산
agent4	Env4: 상태 (S_{t_4})	실제 행동 (A_{t_4})	$\longrightarrow Q(S_{t_4}, A_{t_4})\left(-\sum y_i log p_i\right) \longrightarrow$	Gradient 계산

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

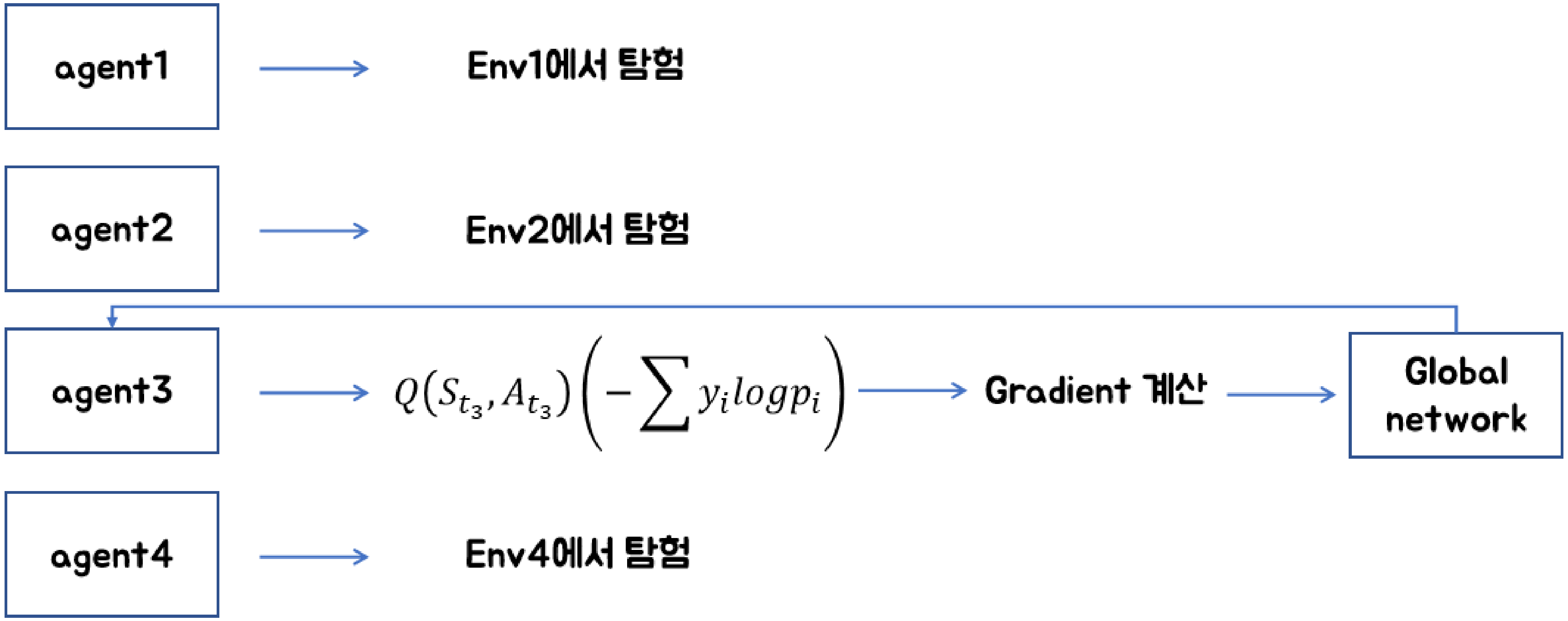
- 비동기적으로 global network를 업데이트



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

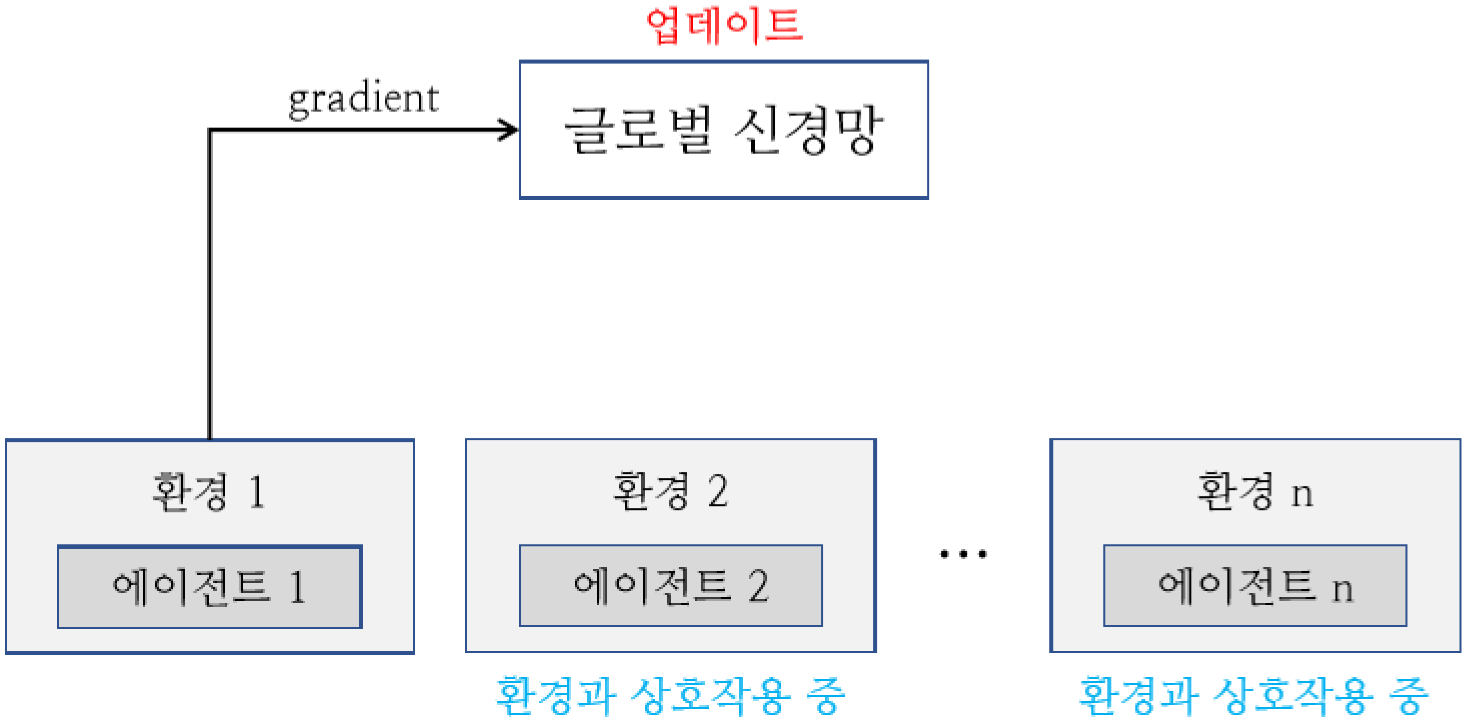
- 비동기적으로 global network를 업데이트



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

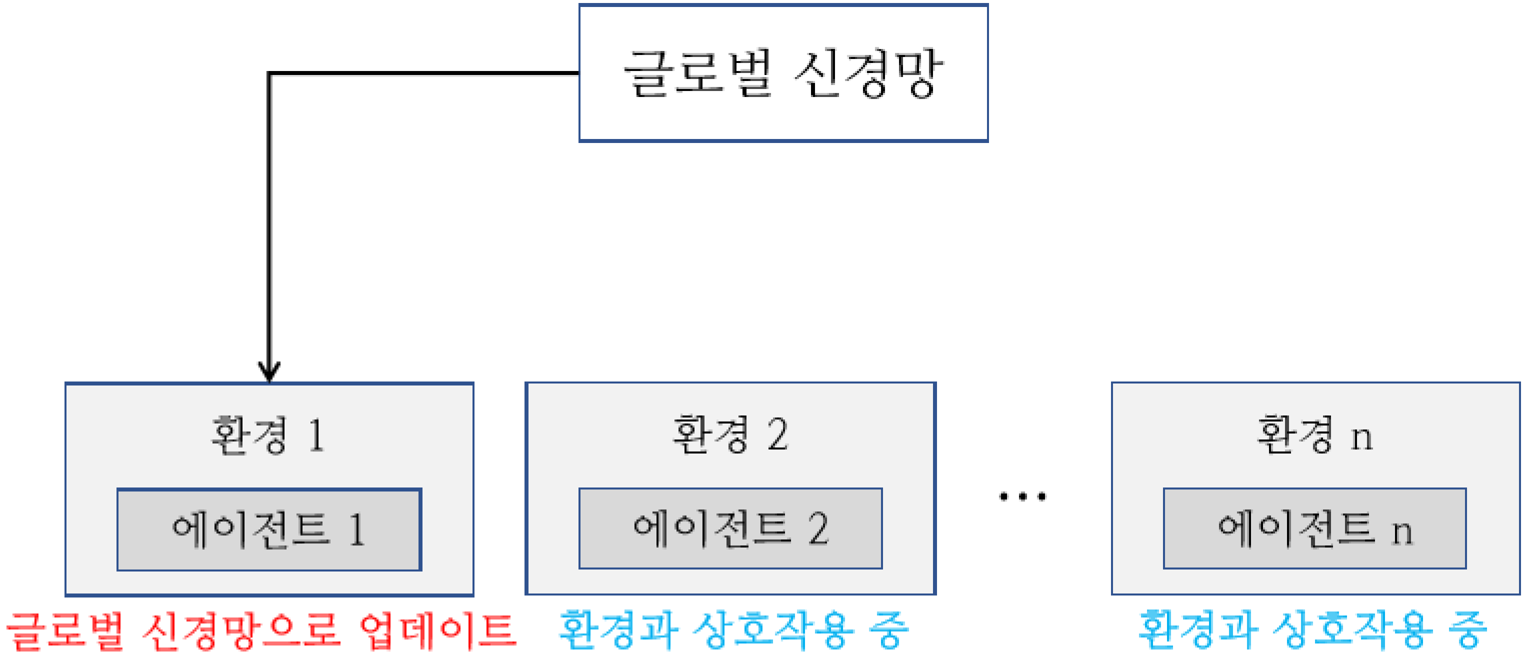
- 비동기적으로 global network를 업데이트: 에이전트1이 글로벌 신경망을 업데이트



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

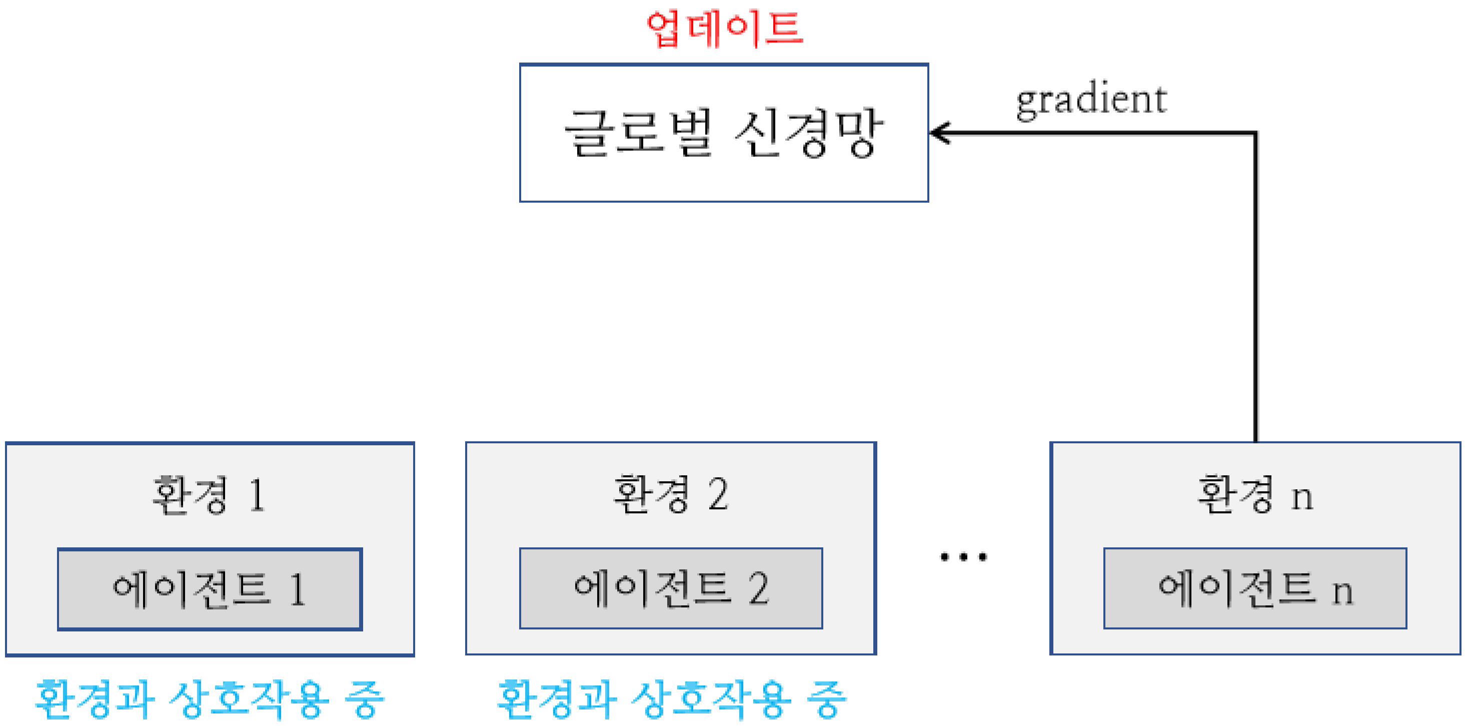
- 비동기적으로 global network를 업데이트: 글로벌 신경망으로 에이전트 1을 업데이트



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

- 비동기적으로 global network를 업데이트: 에이전트n이 글로벌 신경망을 업데이트



이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C : Actor-Critic과 다른 점

Actor를 업데이트하는 과정에서

1. Multi-step loss function

2. Entropy loss function

A3C

1. **Multi-step loss function**

$$-\nabla_\theta log\pi_\theta(a_t|s_t)\underline{(r_{t+1}+\gamma V_v(s_{t+1})-V_v(s_t)\ )}$$

A2C의 Advantage 함수

↓

$$-\nabla_\theta log\pi_\theta(a_t|s_t)\underline{(r_{t+1}+\gamma r_{t+2}+\gamma^2 V_v(s_{t+2})-V_v(s_t)\ )}$$

↓ K-타임스텝 Advantage 함수로 확장

$$-\nabla_\theta log\pi_\theta(a_t|s_t)\underline{(r_{t+1}+\gamma r_{t+2}+\cdots+\gamma^{19} V_v(s_{t+20})-V_v(s_t)\ )}$$

1-step

↓

multi-step

20 step을 가본 후에 loss function이 하나 나온다?

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C

1. Multi-step

$$loss\ funtion = \nabla_\theta log\pi_\theta(a_t|s_t)(r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{19} V_v(s_{t+20}) - V_v(s_t)\ )$$

$$+ \nabla_\theta log\pi_\theta(a_{t+1}|s_{t+1})(r_{t+2} + \gamma r_{t+3} + \cdots + \gamma^{18} V_v(s_{t+20}) - V_v(s_{t+1})\ )$$

$$+ \nabla_\theta log\pi_\theta(a_{t+2}|s_{t+2})(r_{t+3} + \gamma r_{t+4} + \cdots + \gamma^{17} V_v(s_{t+20}) - V_v(s_{t+2})\ )$$

$$+ \cdots +$$

$$\nabla_\theta log\pi_\theta(a_{t+19}|s_{t+19})(r_{t+19} + \gamma V_v(s_{t+20}) - V_v(s_{t+19})\ )$$

20 step 마다 20개의 loss function을 더한 것으로 업데이트

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

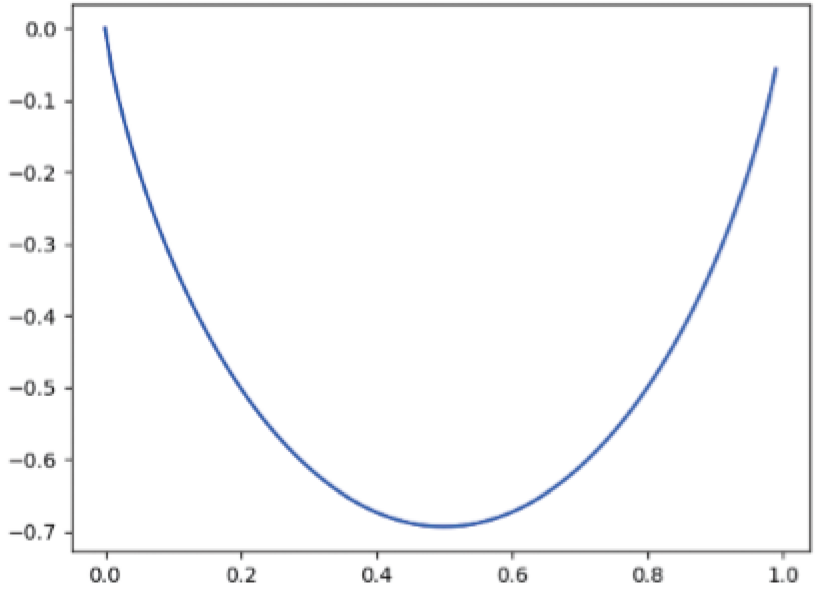
A3C

2. Entropy loss function

엔트로피의 정의: $-\sum_i p_i log p_i$

→ "거꾸로" $\sum_i p_i log p_i$

→ gradient descent 하기 위해!



행동이 두 가지 일 때 행동 확률에 따른 엔트로피의 그래프

A3C

2. Entropy loss function

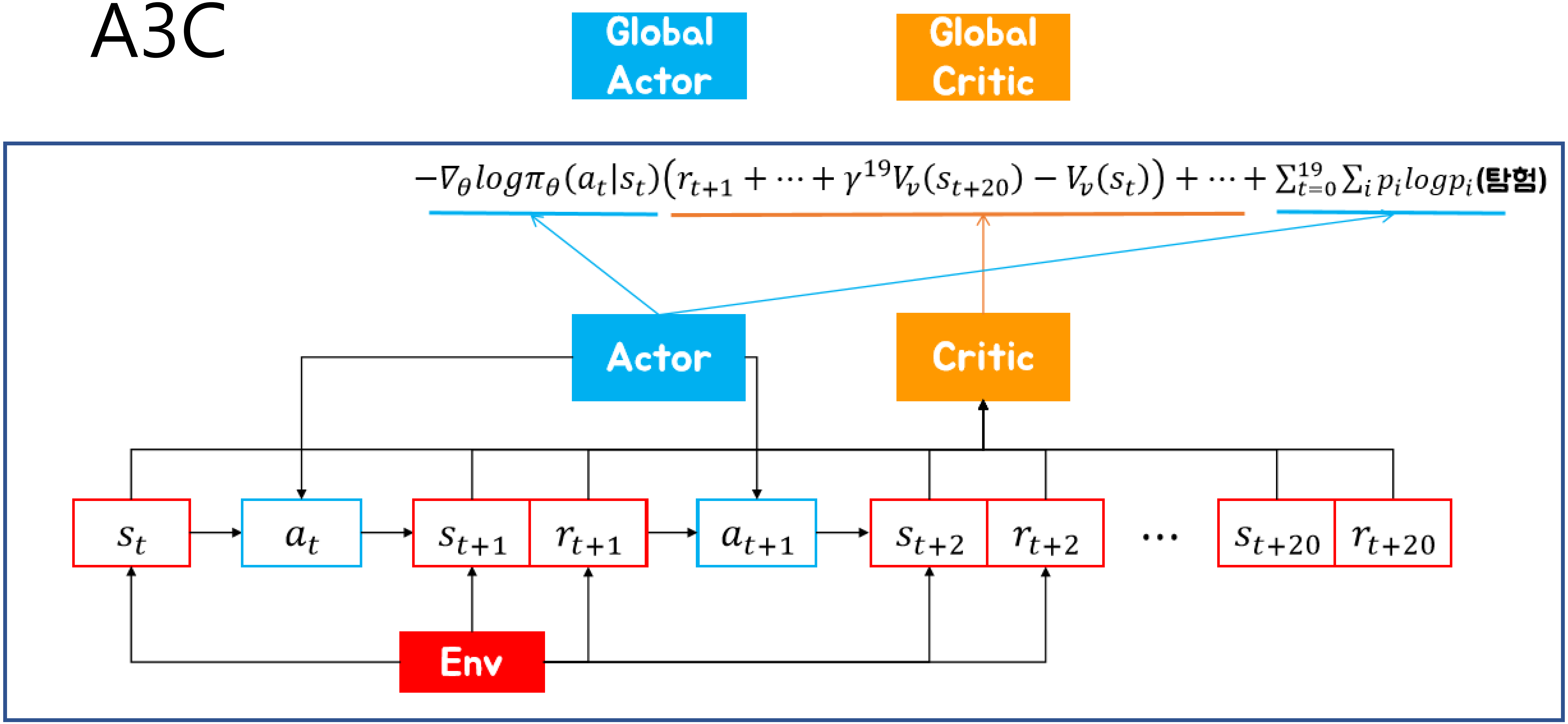
Actor-Critic의 loss function

$$-\nabla_\theta log\pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\ )$$

A3C의 loss function

$$\underbrace{-\nabla_\theta log\pi_\theta(a_{t+19}|s_{t+19})(r_{t+1} + \cdots + \gamma^{19}V_v(s_{t+20}) - V_v(s_t)\ ) \cdots}_{\text{20개의 cross entropy: exploitation}} + \underbrace{\textstyle\sum_{t=0}^{19}\sum_i p_i log p_i}_{\text{Entropy: exploration}}$$

A3C



Thread 1 → 여러 개의 Thread

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT

A3C Summary

- A3C = 비동기 + Actor-Critic
- Actor-Critic과 다른 점
  - Multi-step loss function (cross entropy) -> exploitation
  - Entropy loss function -> exploration

이웅원, RLCode와 A3C 쉽고 깊게 이해하기, PPT