5. REINFORCE

경희대학교

기계공학과 RCI 연구실
박보형

2025-2 이동로봇

목차

- RL vs DRL

- DQN vs PG

- Policy Gradient Theorem

- REINFORCE

RL vs DRL

Reinforcement Learning(RL)

- Tabular Updating Method
  - making state-action value Q(s, a) table
  - finding optimal policy by updating table repeatedly using Bellman Equation

Deep Reinforcement Learning(DRL)

- Approximating the state-action value function or policy by deep neural networks
  - value function Q(s, a) → DQN
  - Policy $\pi(a|s)$ → Policy Gradient (REINFORCE)
  - value function + policy → Actor-Critic (A3C)

DQN vs PG

DQN

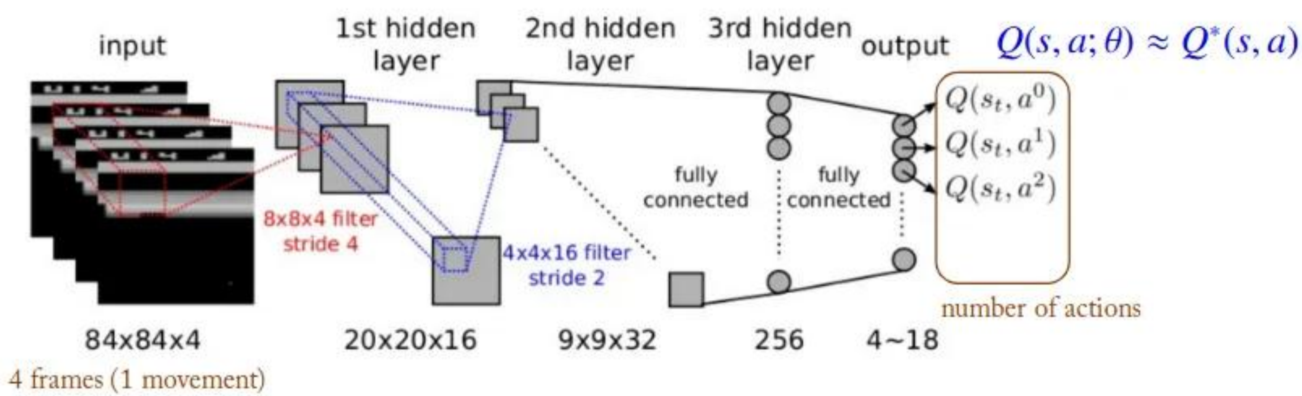
- Q-Learning에서는 State-Action Value Function을 Q-table에 저장해서 사용
- DQN에서는 Q-Table을 CNN(Q-Network)으로 추정하여 사용
    - Input으로 현재 State의 Image Pixel
    - Output으로 모든 Action에 대한 Q값 : 그 중에서 max Q를 갖는 Action을 선택
    - Q-Network를 통해서 Optimal 값을 추정하는 parameter θ를 찾는 것이 학습의 목적
- State 하나만 Input으로 넣기 때문에 State Space의 크기를 고려할 필요가 없다.

  Output의 Node 수는 실제 Action 수만큼 필요하기 때문에 Continuous할 수 없다.

Q-table

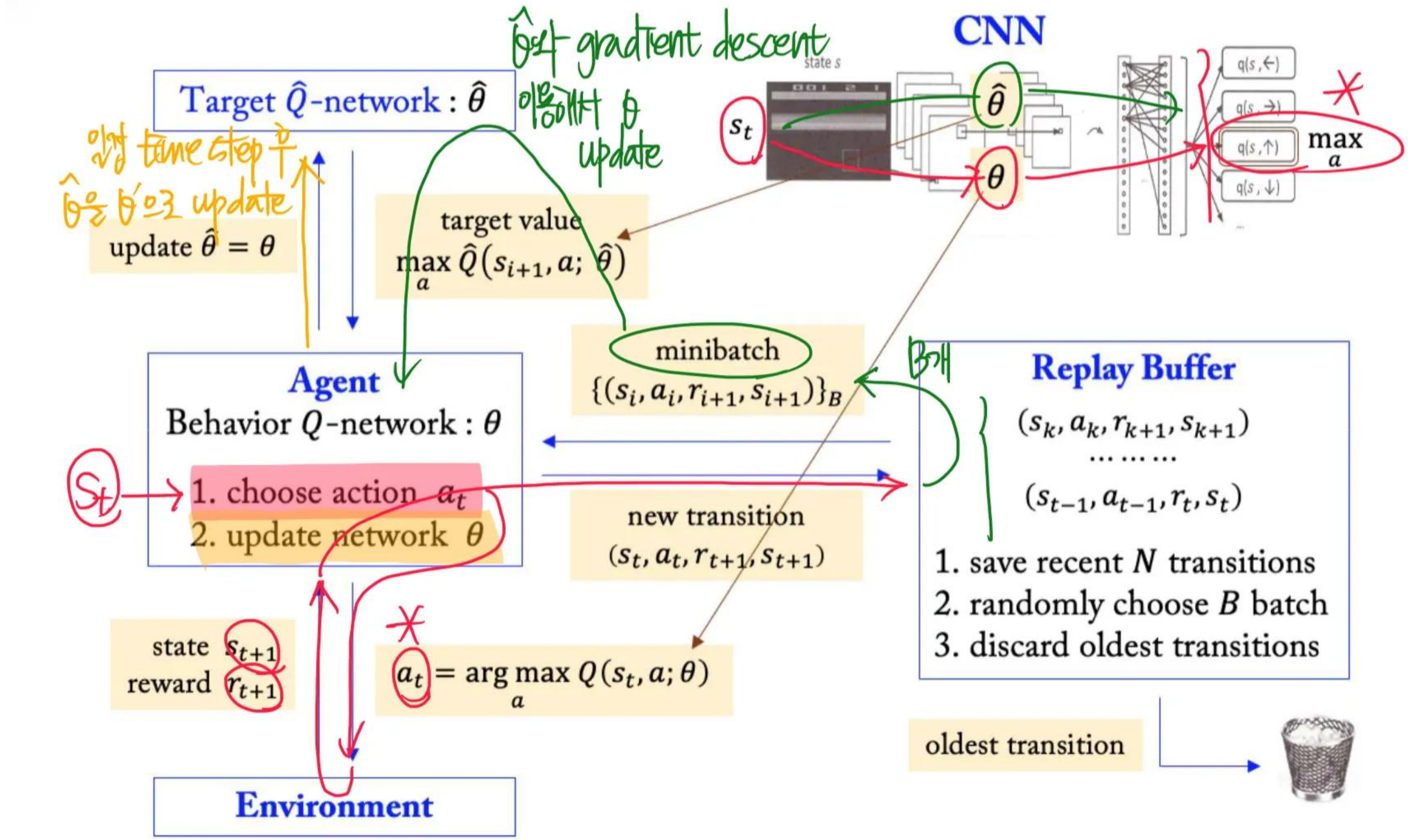
actions

states	$\Uparrow$	$\Downarrow$	$\Leftarrow$	$\Rightarrow$
1	0.49	0.44	0.45	0.41
2	0.40	0.40	0.43	0.42
3	0.48	0.41	0.40	0.29
⋮	⋮	⋮	⋮	⋮
9	0.77	0.57	0.66	0.85



DQN vs PG

DQN



$\hat{\theta}$의 gradient descent
이용해서 θ update
CNN
state s
s_t
$\hat{\theta}$
θ
q(s,←)
q(s,→)
q(s,↑)
q(s,↓)
$\max_a$
Target $\hat{Q}$-network : $\hat{\theta}$
일정 time step 후
$\hat{\theta}$을 θ으로 update
update $\hat{\theta} = \theta$
target value
$\max_a \hat{Q}(s_{i+1}, a; \hat{\theta})$
minibatch
$\{(s_i, a_i, r_{i+1}, s_{i+1})\}_B$
B개
Agent
Behavior Q-network : θ
S_t
1. choose action a_t
2. update network θ
new transition
$(s_t, a_t, r_{t+1}, s_{t+1})$
Replay Buffer
$(s_k, a_k, r_{k+1}, s_{k+1})$
$\cdots\cdots\cdots$
$(s_{t-1}, a_{t-1}, r_t, s_t)$
1. save recent N transitions
2. randomly choose B batch
3. discard oldest transitions
state s_{t+1}
reward r_{t+1}
$a_t = \arg\max_a Q(s_t, a; \theta)$
oldest transition
Environment

DQN vs PG

DQN

Initialize behavior network Q with random weights θ
Initialize target network $\hat{Q}$ with weights $\hat{\theta} = \theta$
Initialize replay buffer $\mathcal{R}$ to capacity N
for episode = 1, M **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocess $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ, select a random action a_t
        otherwise select $a_t = \arg\max_a Q(\phi_t, a; \theta)$ ϵ-greedy CNN θ
        Execute a_t in emulator and observe reward r_{t+1} and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_{t+1}, \phi_{t+1})$ in $\mathcal{R}$
        Sample minibatch of B transitions $(\phi_i, a_i, r_{i+1}, \phi_{i+1})$ from $\mathcal{R}$
        Set $y_i = r_{i+1} + \gamma \max_a \hat{Q}(\phi_{i+1}, a; \hat{\theta})$ CNN $\hat{\theta}$
        Perform a gradient descent on $(y_i - Q(\phi_i, a_i; \theta))^2$ update θ
        Every C steps, reset $\hat{Q} = Q$ (i.e., $\hat{\theta} = \theta$)   update $\hat{\theta}$
    end
end

DQN vs PG

PG

- DQN에서는 Q-Value를 출력하는 Q-Network가 학습 대상
- PG에서는 Action을 직접 출력하는 Policy $\pi_\theta(a|s)$ 자체가 학습 대상
- Object Function(최적화 하려는 대상)

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r(\tau)] = \int p(\tau;\theta)r(\tau)d\tau)$$

τ = trajectory

$r(\tau)$ = epdisode total reward

$p(\tau;\theta)$ = τ가 나올 확률 (pdf)

  - Total Reward를 Maximize하는 방향으로 Update
    - Maximize에는 Gradient Ascent 사용

$$\theta : \theta + \alpha\nabla_\theta J(\theta)$$

    - 그런데 Object Function에 Expectation이 있는데 어떻게 미분해야 하나?

DQN vs PG

PG Theorem

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[r(\tau)] = \mathbb{E}_{\pi_\theta}\left[r(\tau)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

- Object Function의 Gradient는 [total reward]와 [policy에 log를 취한 값을 미분한 것의 sum]과의 product
- 이렇게 했을 때 장점 $J(\theta) = \mathbb{E}_{\pi_\theta}[r(\tau)] = \int p(\tau;\theta) r(\tau) d\tau$
  - 원래는 Expectation을 계산하려면 $p(\tau;\theta)$를 모두 알아야 함 : 불가능
  - Pass 했지만 식 전개 과정에서 $p(\tau;\theta)$가 사라진다.
  - 여전히 Expectation이 있지만 이를 Sampling을 통해서 추정
    - 이를 Markov Chain Monte Carlo(MCMC) 방식이라고 한다.
  - 이렇게 MCMC로 계산한 Total Reward $\mathrm{r}(\tau)$은 varianc가 크다.
    - 그래서 Discounted Return G_t를 사용 = REINFORCE

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

REINFORCE

REINFORCE

Repeat (1) ~ (3)

(1) Execute M trajectories

(each starting in state s and executing (stochastic) policy π_θ)

(2) Approximate the gradient of the objective function $J(\theta)$

$$g_\theta := \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{t=0}^{T-1} G_t^{(i)}\,\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\right) \approx \nabla_\theta J(\theta)$$

(3) Update policy (network parameters) to maximize $J(\theta)$

$$\theta := \theta + \alpha g_\theta \approx \theta + \alpha \nabla_\theta J(\theta)$$

REINFORCE with baseline

Initialize state-value $V(s;\phi)$ and policy $\pi(a \mid s;\theta)$ randomly
Hyperparameters: stepsizes $\alpha > 0,\ \beta > 0$
for episode $= 1, M$ **do**
  Generate an episode $s_0, a_0, r_1, s_1, \ldots, s_{T-1}, a_{T-1}, r_T$, following $\pi(\cdot \mid \cdot;\theta)$
  for $t = 0, T-1$ **do**
    $G_t \leftarrow$ return from step t
    $\delta \leftarrow G_t - V(s_t;\phi)$
    $\phi \leftarrow \phi + \beta\,\delta\,\nabla_\phi V(s_t;\phi)$ minimizing $L(\phi) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1}(G_t - V(s_t;\phi))^2\right]$
    $\theta \leftarrow \theta + \alpha\,\gamma^t \delta\,\nabla_\theta \log \pi(a_t \mid s_t;\theta)$ $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1}(G_t - V(s_t;\phi))\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$
  end
end

- 그런데 Return G_t는 매우 큰 Variance를 가진다 : 학습이 잘 안된다.
- Solution : Return G_t에서 어떤 기준값(baseline)을 빼주고, 그보다 좋은 Action만 강화하자!

Baseline : 현재 상태의 State-Value

$$V(s) = \mathbb{E}_{\pi_\theta}[\,G_t \mid S_t = s\,] \qquad\qquad b(s_t) = V^{\pi}(s_t)$$

- 해당 State에서 평균적으로 얻는 Return을 의미
- Baseline의 State-Value : Critic
- Policy : Actor

Next

REINFORCE 코드 실습

- RL의 Q-Table이 가진 한계를 Q-Network를 통해 State를, PG를 통해 Action을 Continuous로 확장하여 해결 : DRL로의 확장

- REINFORCE 알고리즘을 Cart Pole에 적용한 예제 구현 및 학습 실습

감사합니다