$\rightarrow v, q$.

Value Function Approximation Methods

· 맞춘 생각하지 말고 근사하자. 파라미터 dimension 줄이며

$$\begin{cases} v_\pi(s) \approx \hat{v}(s, w) \text{ where } w \in R^d \text{ with } d \ll |S| \\ q_\pi(s,a) \approx \hat{q}(s,a,w) \text{ where } w \in R^d \text{ with } d \ll |S| \end{cases}$$

- Linear Value Function Approximation

$$J(w) = MSVE(w) = E_\pi[v_\pi(s) - \underbrace{x(s)^T w}_{\hat{v}(s,w)})^2]$$ mean-squared error 사용

only local optimum is possible.

→ Gradient

$$w_{t+1} = w_t + \underbrace{\alpha[G_t - \hat{v}(s_t, w)]\nabla\hat{v}(s_t, w)}_{\Delta w} \quad -\frac{1}{2}\alpha\nabla[v_\pi(s_t) - \hat{v}(s_t, w)]^2$$

$$\text{MC} \quad \Delta w = \alpha(G_t - \hat{v}(s_t, w)]\nabla_w\hat{v}(s_t, w)$$

$$\text{TD(0)} \quad = \alpha(R_{t+1} + \gamma\hat{v}(s_{t+1}, w) - \hat{v}(s_t, w))\nabla_w\hat{v}(s_t, w)$$

$$\text{TD}(\lambda) \quad = \alpha(G_t^\lambda - \hat{v}(s_t, w))(\nabla_w\hat{v}(s_t, w))$$

- Linear Action-Value Function Approximation

$$J(w) = E_\pi[(q_\pi(s,A) - \hat{q}(s,A,w))^2]$$

$$\Delta w = \alpha(q_\pi(s,A) - \hat{q}(s,A,w))x(s,A)$$

$$\text{MC} \quad \Delta w = \alpha(G_t - \hat{q}(s_t, A_t, w))\nabla_w\hat{q}(s_t, A_t, w)$$

$$\text{TD(0)} \quad = \alpha(R_{t+1} + \gamma\hat{q}(s_{t+1}, A_{t+1}, w) - \hat{q}(s_t, A_t, w))\nabla_w\hat{q}(s_t, A_t, w)$$

forward-view
$$\text{TD}(\lambda) \quad = \alpha(q_t^\lambda - \hat{q}(s_t, A_t, w))\nabla_w\hat{q}(s_t, A_t, w)$$

backward-view
$$\text{TD}(\lambda) \quad = \alpha\delta_t E_t$$

$$\delta_t = R_{t+1} + \gamma\hat{q}(s_{t+1}, A_{t+1}, w) - \hat{q}(s_t, A_t, w)$$

$$E_t = \gamma\lambda E_{t-1} + \nabla_w\hat{q}(s_t, A_t, w)$$

+ DQN: Q-learning을 non-linear하게 다루기.

* Off-policy Semi-gradient Learning

$w_{t+1} = w_t +$

TD(0) $\quad \alpha \underline{\rho_t} [R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)] \nabla \hat{v}(S_t, w)$

SARSA(0) $\quad \alpha \underline{\rho_{t+1}} [(R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, w)) - \hat{q}(S_t, A_t, w)] \nabla \hat{q}(S_t, A_t, w)$

n-step SARSA $\quad \alpha \underline{\rho_{t+1:t+n}} [G_{t:t+n} - \hat{q}(S_t, A_t, w_{t+n-1})] \nabla \hat{q}(S_t, A_t, w_{t+n-1})$

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, w_{t+n-1})$$

* Batch Reinforcement Learning. 끝에 업데이트

$D = \{\langle S_1, V_1^\pi \rangle, \langle S_2, V_2^\pi \rangle \cdots \langle S_T, V_T^\pi \rangle\}$ Batch 만큼 계산

$LS(w) = E_D[(V^\pi - \underbrace{\hat{v}(S, w)}_{w^T x(S)})^2]$ 이 최소가 되야하는 w 구하기

① 수식적 : 계산량 ↑↑

② Prediction

LSMC $\rightarrow 0 = \sum_{t=1}^{T} \alpha (G_t - \hat{v}(S, w)) x(S)$

LSTD $\rightarrow 0 = \sum_{t=1}^{T} \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) x(S_t)$

LSTD(λ) $\rightarrow 0 = \sum_{t=1}^{T} \alpha \delta_t E_T$

LSTDQ $\rightarrow 0 = \sum_{t=1}^{T} \alpha (R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}), w) - \hat{q}(S_t, A_t, w)) x(S_t, A_t)$

total update = zero

DQN (Deep Q-Networks)

Q-learning을 Neural Network으로 해석 (table 형태의 학습은 속도가느리다)

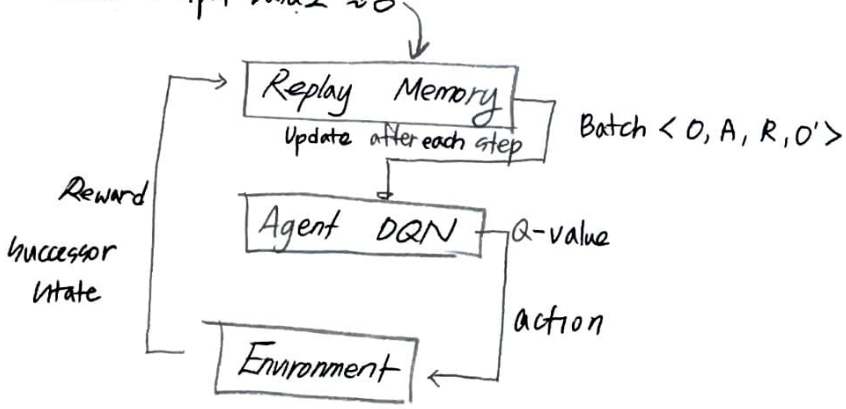
$\langle s, v^{\pi} \rangle \sim D$ 에피소드저장 → sample

$\Delta w = \alpha [v^{\pi} - \hat{v}(s, w)] \nabla_w \hat{v}(s, w)$

$\Rightarrow w^{\pi} = \arg\min_{w} LS(w)$ 개별 state가 중요X
w가 바뀌면 전체가 바뀌기 때문에

→ episode 간
correlation이 존재, 비상관적X
(∵ 몇개의 선택되는 state를 이용
→ sampling)

- state이 input Data로 활용



$L_i(\theta_i) = E_{s,a \sim p(\cdot)} [(y_i - Q(s, a; \theta_i))^2]$ 을 바탕으로 Neural Network에 넣기

mini-batch data 에 대해 bootstrap