# CH12



## 12.1 강화학습이란

- 어떤 환경에서 어떤 행동을 했을 때 그것이 잘된 행동인지 잘못된 행동인지를 판단하고 보상(또는 벌칙)을 주는 과정을 반복해서 스스로 학습하게 하는 분야


▲ 그림 12-1 강화 학습

- 환경 : 에이전트가 다양한 행동을 해 보고, 그에 따른 결과를 관측할 수 있는 시뮬레이터
- 에이전트 : 환경에서 행동 주체

## 12.2 마르코프 결정 과정

- 강화 학습 = 마르코프 결정 과정에서 학습 개념 추가

### 마르코프 프로세스

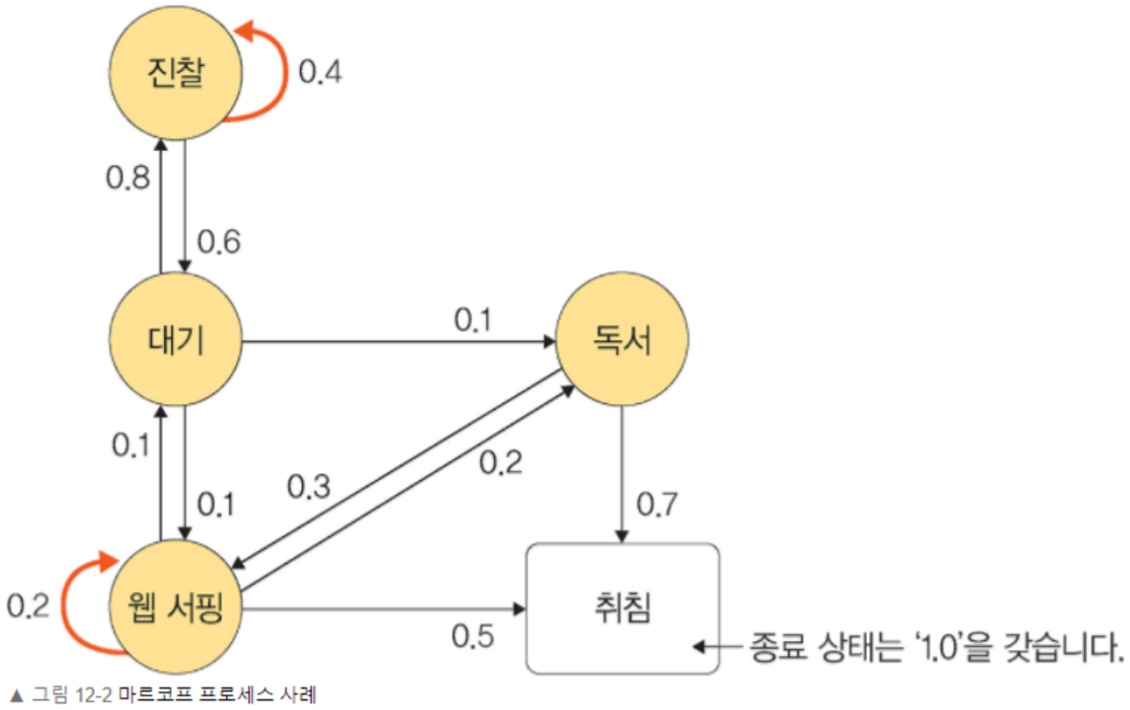
: 어떤 상태가 일정 간격으로 변하고, 다음 상태는 현 상태에만 의존하는 확률적 상태 변화 (미래 상태는 과거 상태와는 독립적, 현재 상태로만 결정)

- 마르코프 특성을 지니는 이산 시간에 대한 확률 과정
- 마르코프 체인은 시간에 따른 상태 변화를 나타내며, 상태 변화를 **전이**라고 함 → 상태 전이  확률

$$P(S_{t+1} = s' \mid S_t = s) = P_{ss'}$$

($S$: 상태의 집합

$P_{ss'}$: 상태의 변화를 확률로 표현)




▲ 그림 12-2 마르코프 프로세스 사례

### 마르코프 보상 프로세스(MRP)

: 각 상태마다 좋고 나쁨이 추가된 확률 모델(보상이나 벌칙)

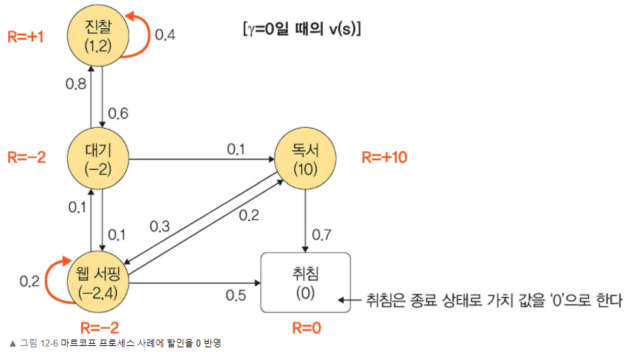
- 리턴 : 모든 보상의 합

$$G_t = \underbrace{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots}_{k+1 \text{ 시점 이후부터 } \gamma^k \text{ 적용}} = \sum_{k=1}^{\infty} \gamma^k r_{t+k+1}$$

⇒ 할인율이 적용된 리턴 수식

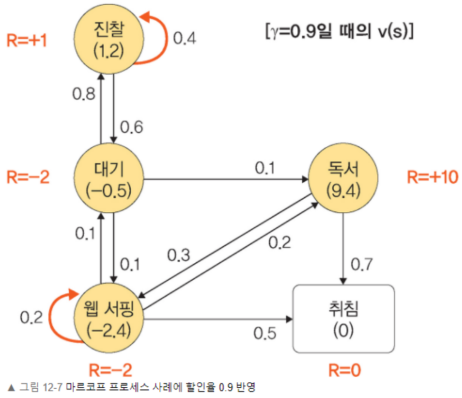
- 가치 함수 : 기대되는 모든 보상의 합 = 가치

**⇒ 미래 가치가 최대되는 결정을 하고 행동하는 것이 강화 학습의 목표**



▲ 그림 12-6 마르코프 프로세스 사례에 할인율 0 반영

독서 가치 = 10 + 0×[(-2.4×0.3) + (0×0.7)] = 10



▲ 그림 12-7 마르코프 프로세스 사례에 할인율 0.9 반영

독서 가치 = 10 + 0.9×[(-2.4×0.3) + (0×0.7)] = 9.4

- 할인율이 1이라면 미래 가치의 할인을 고려하지 않으므로 원시안적 상태로 가치 값이 나타남!

## 마르코프 결정 과정(MDP)

: 마르코프 보상 과정에 행동 추가된 확률 모델

- 각 상태마다 전체 보상 최대화하는 행동 결정
- 정책 : 행동 분포을 표현하는 함수

- **상태-가치 함수** : 상태에서 얻을 수 있는 리턴 기댓값
- **행동-가치 함수** : 행동을 취했을 때 얻는 리턴 기댓값

$$
\begin{aligned}
v_\pi(s) &= E_\pi[G_t \mid S_t = s] \\
&= E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s] \\
&= E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= E_\pi[\underbrace{R_{t+1}}_{①} + \underbrace{\gamma v_\pi(S_{t+1})}_{②} \mid S_t = s]
\end{aligned}
$$

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi[G_t \mid S_t = s, A_t = a] \\
&= E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a]
\end{aligned}
$$

- O(n^3) 해결을 위해 다이나믹 프로그래밍/몬테카를로/시간차학습/함수적 접근방법이 있음

  1. 다이나믹 프로그래밍(dynamic programming)

  : 마르코프 결정 과정의 상태와 행동이 많지 않고 모든 상태와 전이 확률을 알고 있다면 다이나믹 프로그래밍 방식으로 각 상태의 가치와 최적의 행동을 찾을 수 있습니다. 하지만 대부분의 강화 학습 문제는 상태도 많고, 상태가 전이되는 경우의 수도 많으므로 다이나믹 프로그래밍을 적용하기 어렵습니다.

  2. 몬테카를로(Monte Carlo method)

  : 마르코프 결정 과정에서 상태가 많거나 모든 상태를 알 수 없는 경우에는 다이나믹 프로그래밍을 적용할 수 없었습니다. 몬테카를로는 전체 상태 중 일부 구간만 방문하여 근사적으로 가치를 추정합니다. 초기 상태에서 시작하여 중간 상태들을 경유해서 최종(terminal) 상태까지 간 후 최종 보상을 측정하고 방문했던 상태들의 가치를 업데이트합니다.

  3. 시간 차 학습(temporal difference learning)

  : 몬테카를로는 최종 상태까지 도달한 후에야 방문한 상태들의 업데이트가 가능하다는 단점이 있습니다. 시간 차 학습 방식은 최종 상태에 도달하기 전에 방문한 상태의 가치를 즉시 업데이트합니다. 즉, 시간 차 학습은 다이나믹 프로그래밍과 몬테카를로의 중간적인 특성을 가지며 본격적인 강화 학습의 단계라고 할 수 있습니다.

  4. 함수적 접근 학습(function approximation learning)

  : 마르코프 결정 과정의 상태가 아주 많거나, 상태가 연속적인 값을 갖는 경우는 상태-가치 함수나 행동-가치 함수를 테이블 형태로 학습하기 어렵습니다. 함수적 접근 학습 방법은 연속적인 상태를 학습하고자 상태와 관련된 특성 벡터를 도입했습니다. 특성의 가중치를 업데이트하여 가치의 근사치를 찾을 수 있습니다.

## 12.3 MDP를 위한 벨만 방정식

: 상태-가치/행동-가치 함수 관계를 나타냄

## 벨만 기대 방정식

- 정책을 고려해 다음 상태로 이동
- 재귀적 형태로서 미래 가치들이 현재 가치에 영향을 주고 있는 형태
  - **상태-가치**

$$
\begin{aligned}
v_\pi(s) &= E_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \sum_{a \in A} \pi(a|s) \sum_{s',r} P(s', r | s, a)[r + \gamma E_\pi[G_{t+1} | S_{t+1} = s']] \\
&= \sum_{a \in A} \pi(a|s) \sum_{s',r} P(s', r | s, a)[r + \gamma v_\pi(s')] \cdots ③ \\
&= \sum_{a \in A} \pi(a|s) q_\pi(s,a) \cdots ④
\end{aligned}
$$

$$
v_\pi(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s'))
$$

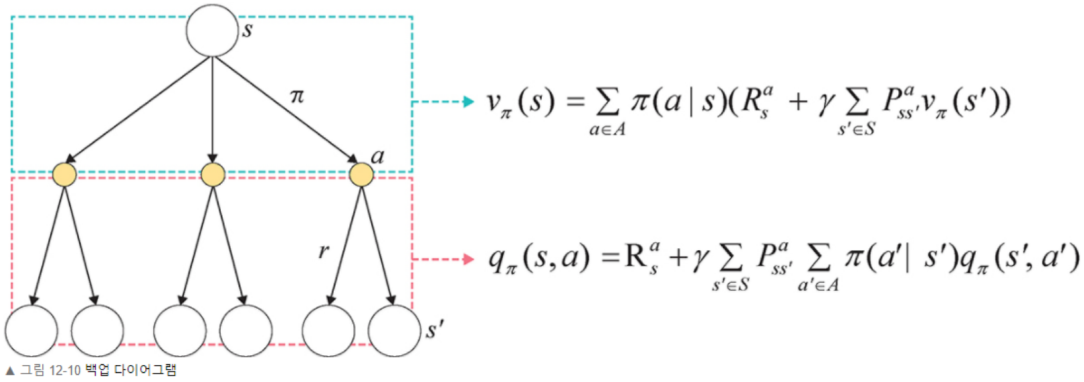
  - **행동-가치**

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi[G_t | S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma E_\pi(G_{t+1} | S_{t+1} = s') | S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma E(G_{t+1} | S_{t+1} = s', A_{t+1} = a') | S_t = s, A_t = a] \\
&= E_\pi[R_{t+1} + \gamma v_\pi(s') | S_t = s, A_t = a] \\
&= \mathrm{R}_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a')
\end{aligned}
$$

  - **최종⭐**

$$
v_\pi(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s'))
$$

$$
q_\pi(s,a) = \mathrm{R}_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a')
$$



▲ 그림 12-10 백업 다이어그램

⇒ 상태-가치 함수 vπ(s)는 뒤따를 행동-가치 함수의 정책 기반 가중 평균으로 이해하면 되고, 행동-가치 함수 qπ(s,a)는 다음 상태-가치 함수에 대한 보상과 상태 전이 확률에 대한 결합 확률의 가중 평균으로 이해

⇒ 미래의 가치 함수 값들을 이용하여 초기화된 임의의 값들을 업데이트하면서 최적의 가치 함수로 다가가는 것입니다. 즉, 강화 학습은 가치 함수 초깃값(0 혹은 랜덤한 숫자들)을 2~3의 과정을 반복하며 얻은 정보들로 업데이트하여 최적의 값을 얻기

## 벨만 최적 방정식

: 강화 학습 목표에 따라 찾은 최정화된 정책을 따르는 벨만 방정식

**최적의 상태-가치 함수(optimal state-value function)**

$$v_*(s) = \max_\pi v_\pi(s)$$

**최적의 행동-가치 함수(optimal action-value function)**

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

## 다이나믹 프로그래밍

: 연속적으로 발생하는 문제를 수학적으로 최적화

- MDP의 모든 상황 알고 있다고 가정 → **계획**

: 정책 업데이트

- 발전 : 가치 값을 비교하여 더 좋은 정책 찾기
- 평가 : 현 정책을 이용하여 가치 함수 찾기

⇒ 수렴하는 부분이 최적화 부분

**정책 이터레이션**

1) 정책 평가 : 모든 상태에 대해 다음 상태가 될 수 있는 행동에 대한 보상 합을 저장

2) 정책 발전

- 욕심쟁이 정책 발전 : 행동-가치 함수 최대값 행동을 선택

**가치 이터레이션**

: 최적 정책 가정하고 벨만 최적 방정식으로 순차적 행동 결정 → 정책 발전 필요 없음

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s,\ A_t = a]$$

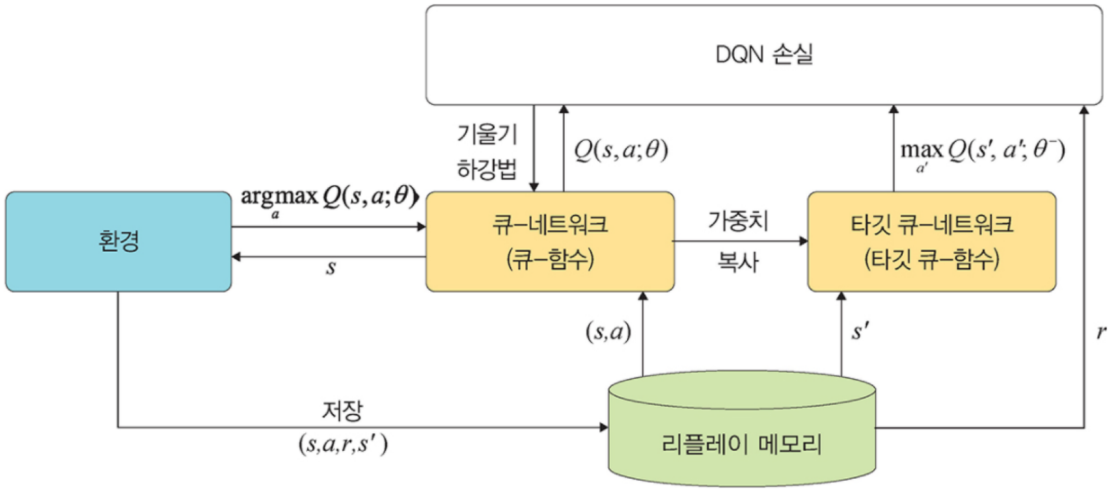
## 12.4 큐-러닝

- 모델 없이 학습하는 강화 학습
- 에이전트가 행동을 취했을 때 받는 보상 기댓값 예측하는 큐함수
- 실험을 통해 최적 정책 학습(행동 랜덤, 새로운 길= 탐험(시간, 자원 낭비일수도), 욕심쟁이 방법으로)
- **경험으로 학습하는 강화 학습에서 최단 시간에 주어진 환경의 모든 상태를 관찰하고, 이를 기반으로 보상을 최대화할 수 있는 행동을 수행하려면 활용(exploitation)과 탐험(exploration) 사이의 균형이 필요**
- 단점
  - 에이전트가 취할 수 있는 상태 개수가 많은 경우 큐-테이블 구축 한계
  - 데이터 간 상관관계로 학습이 어렵

### DQN(딥 큐러닝)

: 합성곱 신경망 이용해서 큐함수 학습. 층 깊게하여 큐 값 정확도 높이기

- 타깃 큐 네트워크 : 큐 값 바뀌지 않도록 큐 네트워크 외 별도 타깃 큐 네트워크를 두고, 수렴을 원활히 하기 위해 주기적 업데이트함

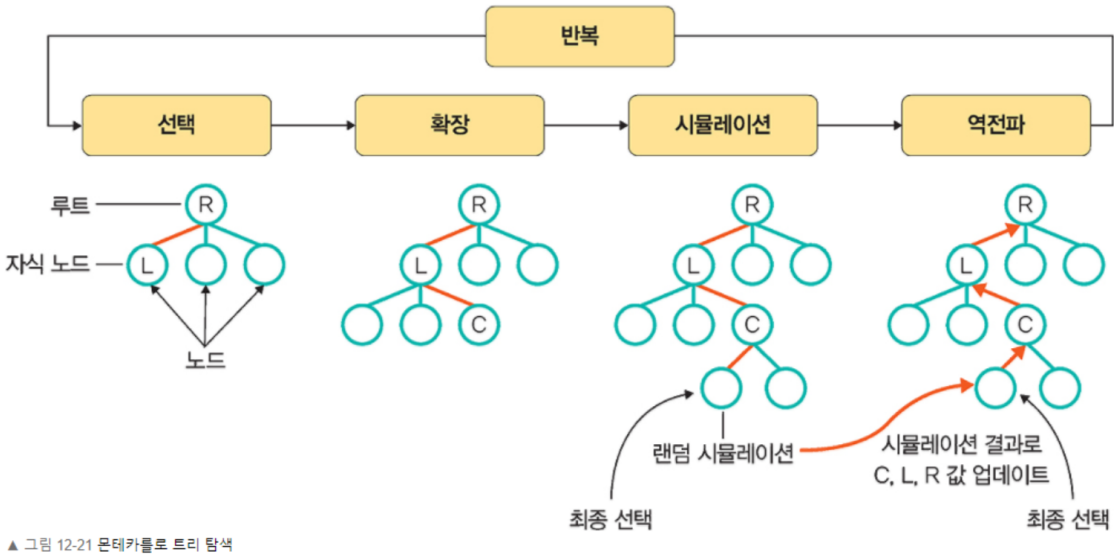



▲ 그림 12-14 딥 큐-러닝 네트워크 상세 구조

- 손실함수 MSE
- 리플레이 메모리 : 일정 데이터 수집될 때까지 기다리다가 랜덤으로 추출하여 미니 배치를 통해 훈련하므로 상관관계 문제 해결

## 12.5 몬테카를로 트리 탐색

: 게임 시뮬레이션을 이용하여 가능성이 높은 방향으로 행동 결정(순차적이 아닌 승률 높은 값 시도)




▲ 그림 12-21 몬테카를로 트리 탐색

- 예) 알파고, 보드게임/포커 등의 비결정적 게임