

MDP와 벨만 방정식

2017

Link Lab 겨울 세미나
12th 김영규

개요

- 강화학습은 어떠한 방정식을 풀어내는 방법.
 - 이 방정식이 벨만 방정식.
-
- 순차적 행동 결정 문제는 MDP로 정의할 수 있다.
 - MDP를 통해 정의된 문제에서 에이전트가 학습하기 위해 가치함수라는 개념을 도입한다.
 - 가치함수는 벨만 방정식과 연결된다.

MDP

- 강화학습은 순차적으로 행동을 계속 결정해야 하는 문제를 푸는 것이다.
- 이러한 문제를 **수학적으로 표현**한 것이 MDP이다.
- MDP는 '**상태, 행동, 보상함수, 상태 변환 확률, 감가율**'로 구성된다.

MDP의 구성요소

상태

행동

보상 함수

상태 변환 확률




감가율

MDP

- 사람이 문제를 풀기 위해서는 먼저 어떤 문제인지 파악하고 스스로 문제를 정의한다.
- 강화학습에서는 **사용자가 문제를 정의**해야 한다.
- 문제를 잘못 정의하면 에이전트가 학습을 못할 수 있다.
- 문제 정의는 에이전트가 학습하는데 **가장 중요한 단계 중 하나**이다.
- 에이전트를 구현하는 사람은 **학습하기에 많지도 적지도 않은 적절한 정보**를 에이전트가 알 수 있도록 문제를 정의해야 한다.

MDP

- 순차적으로 행동을 결정하는 문제에 대한 정의 = MDP
- 예) 그리드월드(Grid World)
 - 그리드(Grid) : 격자
 - 그리드월드 : 격자로 이루어진 환경
 - MDP는 이런 환경을 컴퓨터가 이해할 수 있도록 재정의한다.

(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)
(1, 2)	(2, 2)	 R: -1.0	(4, 2)	(5, 2)
(1, 3)	 R: -1.0	 R: 1.0	(4, 3)	(5, 3)
(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)
(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)

* 그리드 월드

상태

- S 는 에이전트가 관찰 가능한 상태의 집합.
- 상태 = “자신의 상황에 대한 관찰”
 - 예) 로봇의 센서 값
- 게임을 학습하기 위한 에이전트는 **사용자가 상태를 정의**.
- 상태를 정의할 때 에이전트가 학습하기에 **충분한 정보**를 주어야 한다.
- 그리드월드에서는 상태 공간이 작으므로 상태의 정의 문제가 중요하지 않을 수 있다.
- 방대하고 복잡한 상태 안에서 학습하는 에이전트는 상태 정의 문제가 중요하다.

$$S = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

* 상태의 집합

상태

- 에이전트는 시간에 따라 상태 집합의 상태를 탐험한다.
- 시간을 t 라고 표현하고 t 일 때 상태를 s_t 라고 표현한다.
- 어떤 t 에서의 상태 s_t 는 정해져 있지 않다.

$$s_t = (1,3)$$

* 시간 t 에 $(1,3)$ 이라는 상태





- 이처럼 어떤 집합 안에서 뽑을 때마다 달라질 수 있는 것을 “확률변수” 라고 한다.
 - 예) 주사위 던지기, 동전 던지기 등
- 확률변수는 대문자로 표현한다.
- 따라서 시간 t 에서 상태를 s_t 와 같이 대문자로 쓴다.

$$s_t = s$$

* 시간 t 에서의 상태 s_t 가 어떤 상태 s 다.

그리드월드에서의 상태

- 그리드월드에서는 격자 상의 각 위치(좌표)가 상태.
 - 상태는 총 25개의 유한한 개수.
- 빨간색 네모는 에이전트
- 에이전트가 (1, 1)에 있으면 에이전트의 상태는 (1, 1)

 (1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)
(1, 2)	(2, 2)	R: -1.0  (3, 2)	(4, 2)	(5, 2)
(1, 3)	R: -1.0  (2, 3)	R: 1.0  (3, 3)	(4, 3)	(5, 3)
(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)
(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)

* 그리드 월드

$$S = \{(1,1), (1,2), (1,3), \dots, (5,5)\}$$

* 그리드월드의 상태 집합

행동

- 에이전트가 상태 s_t 에서 할 수 있는 가능한 행동의 집합 A .
- 보통 에이전트가 할 수 있는 행동은 **모든 상태에서 같고** 따라서 하나의 집합 A 로 나타낼 수 있다.

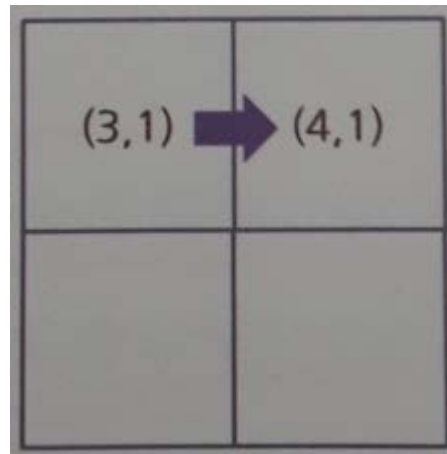
$$A_t = a$$

* 시간 t 에서의 행동 a

- A_t 는 어떤 t 라는 시간에 집합 A 에서 선택한 행동이다.
- t 에서 **어떤 행동을 할 지 정해져 있는 것이 아니므로**(확률변수) A_t 와 같이 대문자로 표현한다.
- 보통 행동집합은 한 문제 내에서 **변하지 않는다**.

그리드월드에서의 행동

- 그리드월드에서 에이전트가 할 수 있는 행동 : up, down, left, right
 $A = \{up, down, left, right\}$
- 시간 t 에서 상태가 $(3,1)$ 이고 $A_t = right$ 라면 $t+1$ 에서의 상태는 $(4,1)$



* $(3,1)$ 상태에서 right행동을 한 후 에이전트가 이동

보상함수

- 보상
 - 에이전트가 학습할 수 있는 유일한 정보
 - 환경이 에이전트에게 주는 정보

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

* 보상함수의 정의

- 보상함수는 시간 t 일 때 상태가 $S_t = s$ 이고 그 상태에서 행동 $A_t = a$ 를 했을 경우에 받을 **보상에 대한 기댓값** E 이다.
- 즉, R_s^a 는 보상의 기댓값. (받을 것이라 예상되는 숫자)

보상함수

- 기댓값
 - 일종의 평균
 - 어떤 정확한 값이 아니라 나오게 될 숫자에 대한 예상
 - 기댓값은 대문자 E(Expectation)로 표시
- 보상함수를 기댓값으로 표현하는 이유
 - 보상을 에이전트에게 주는 것은 환경
 - 환경에 따라서 같은 상태에서 **같은 행동을 취하더라도 다른 보상을 줄 수 있음.**
- 조건부 확률의 표현
 - 보상함수의 괄호 안의 "|"는 조건문에 대한 표현이다.
 - "|"의 뒷 부분은 현재의 조건(=상태와 행동)을 의미.

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

* 보상함수의 정의

t+1에서 받는 보상

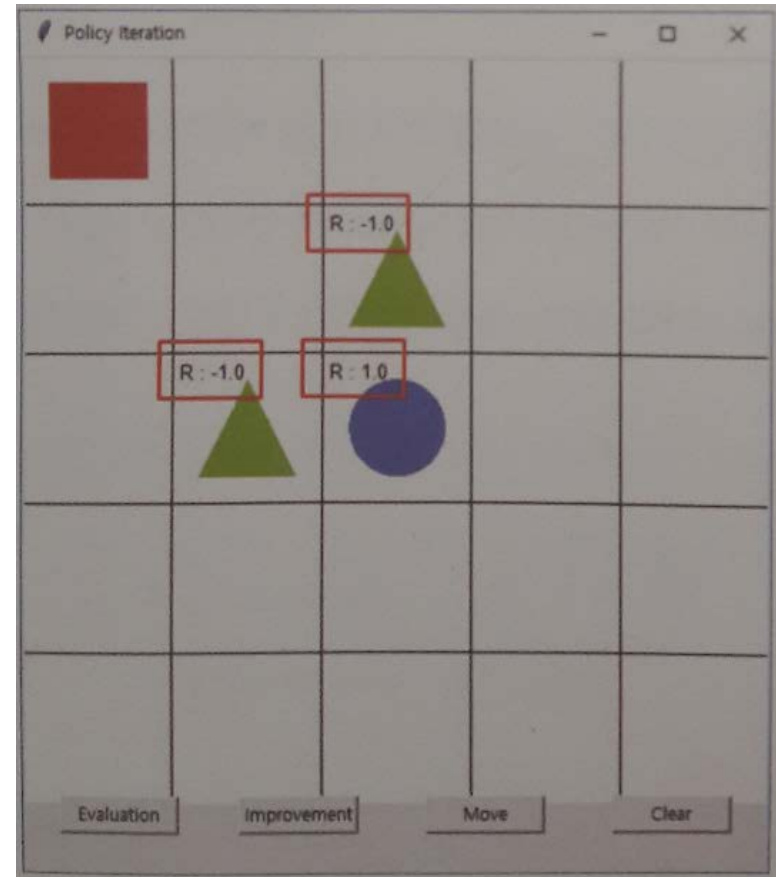
- 어떤 상태에서 행동한 것은 시간 t, 보상을 받는 것은 t+1
- t+1에서 보상을 받는 이유
 - 보상을 에이전트가 알고 있는 것이 아니다.
 - 보상은 **환경이 알려주는 것이다.**
 - 에이전트가 상태 s에서 행동 a를 하면 환경은 에이전트가 가게 되는 다음 상태 s' 과 에이전트가 받을 보상을 에이전트에게 알려준다.
 - 환경이 에이전트에게 **알려주는 시점은 t+1**인 시점이다.
 - 에이전트가 받는 보상 = R_{t+1}

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

* 보상함수의 정의

그리드월드에서의 보상

- 파란색 동그라미가 있는 상태로 행동했을 때, (+1) 보상
- 초록색 세모가 있는 상태로 행동했을 때, (-1) 보상
- 에이전트는 하나의 타임스텝 (시간 단위)이 지난 다음 타임스텝에 보상을 받는다.
- 에이전트는 파란색 동그라미로 가는 "좋은 행동"을 했기 때문에 보상을 받는다.



* 그리드월드에서의 보상

상태 변환 확률

- 상태 s 에서 행동 a 를 취했을 때 다른 상태 s' 에 도달할 확률.
- 보상과 마찬가지로 **에이전트가 알지 못하는 값. 환경의 일부.**
- 환경의 모델이라고도 부른다.
- 상태 변화에는 확률적인 요인(바람이 불거나 넘어짐 등)이 들어가고 이를 수치적으로 표현한 것이 상태변환 확률이다.
- 환경은 에이전트가 행동을 취하면 상태 변환 확률을 통해 다음에 에이전트가 갈 상태를 알려준다.
- 아래 수식의 괄호 앞의 P 는 확률을 의미.

$$P_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$$

* 상태 변환 확률

감가율

- 에이전트는 항상 **현재**에 판단을 내린다.
- 현재에 가까운 보상일 수록 더 큰 가치다.
- 같은 보상이면 나중에 받을수록 가치가 줄어든다.
 - 예) 현실 세계의 이자
- 에이전트는 보상이 얼마나 시간이 지나서 받는지를 고려해서 감가시켜 현재의 가치로 따진다.
- 시간에 따라서 감가하는 비율을 수학적으로 표현하기 위해 “감가율”이라는 개념을 도입한다.
- 감가율은 γ 로 표기하고 γ 는 0과 1 사이의 값이다.
- 현재의 시간 t 로부터 시간 k 가 지난 후에 보상 R_{t+k} 을 받을 것이라면 시간이 k 만큼 지났기 때문에 R_{t+k} 은 γ^{k-1} 만큼 감가된다.

$$\gamma \in [0,1]$$

* 감가율의 정의

$$\gamma^{k-1} R_{t+k}$$

* 감가율을 고려한 미래 보상의 현재 가치

정책

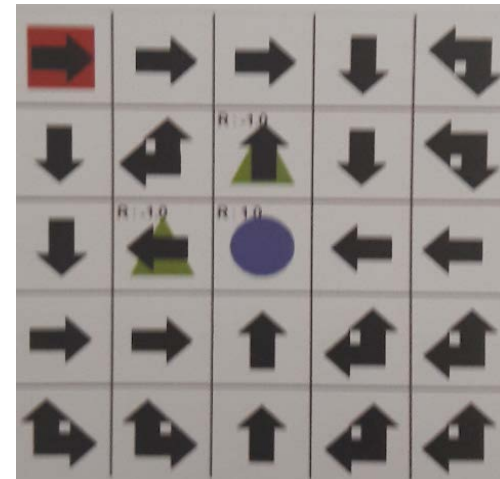
- 정책은 모든 상태에서 에이전트가 할 행동.
- 각 상태에서 단 하나의 행동만을 나타낼 수 있다.
- 에이전트가 **강화학습을 통해 학습해야 할 것은 최적 정책**이다.
- 최적 정책은 각 상태에서 단 하나의 행동만을 선택한다.
- 에이전트가 학습하고 있을 때는 **확률적으로 여러 개의 행동을 선택** 할 수 있어야 한다. **다양한 상황에 대해 학습**하고 최적 정책을 찾을 수 있다.

정책

$$\pi(a|s) = P[A_t = a|S_t = s]$$

* 정책의 의미

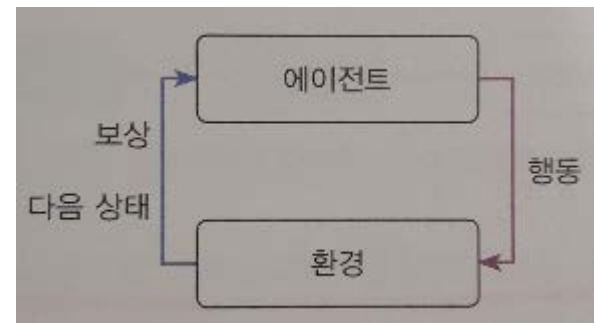
- 정책의 의미는 '시간 t 에 $s_t = s$ 에 에이전트가 있을 때 가능한 행동 중에서 $A_t = a$ 를 할 확률' 이다.
- 정책으로 모든 상태에 에이전트가 해야 할 행동을 알 수 있다.
- 강화학습은 "최적 정책"을 얻기 위해 **현재보다 더 좋은 정책을 학습**해나가는 것이다.



* 그리드월드에서 정책의 예

최적 정책 학습 과정

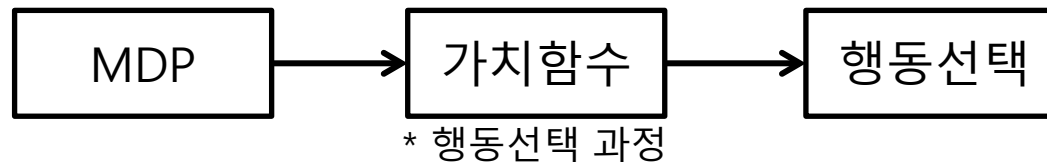
1. MDP로 순차적 행동 결정 문제를 정의한다.
2. 에이전트는 현재 상태에서 앞으로 받을 보상을 고려해서 행동을 결정한다.
3. 환경은 에이전트에게 실제 보상과 다음 상태를 알려준다.
4. 2. 와 3.을 반복하며 가치함수(앞으로 받을 것이라 예상했던 보상)이 틀렸다는 것을 알게 된다.
5. 에이전트는 실제 받은 보상을 토대로 자신의 정보와 정책을 바꾸어 나간다.
6. 위의 학습 과정을 무한히 반복하면 가장 많은 보상을 받게 하는 정책을 학습할 수 있다.



* 에이전트와 환경의 상호작용

가치함수

- 어떤 행동을 할지 좋은 선택하기 위해 앞으로 받을 보상들을 고려해야 한다.
- 가치함수란 아직 받지 않은 보상들을 고려하기 위한 개념으로, 가치함수의 정의는 "현재 상태에서부터 정책을 따라갔을 때 받을 것이라 예상되는 보상의 합"이다.
- 즉, 가치함수란 에이전트가 **어떤 정책이 더 좋은 정책인지 판단하는 기준**이다.



가치함수

- 보상은 행동을 한 다음 타임스텝에서 받는다.
- 시간 t 에 행동을 해서 받는 보상은 R_{t+1} 이고, $t+1$ 에 행동을 해서 받는 보상은 R_{t+2} 이다. 아래의 수식은 시간마다 받는 보상의 합이다.

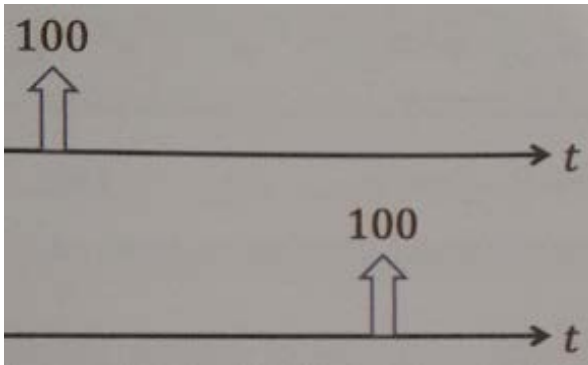
$$R_{t+1} + R_{t+2} + R_{t+3} + R_{t+4} + R_{t+5} + \dots$$

* 단순한 보상의 합

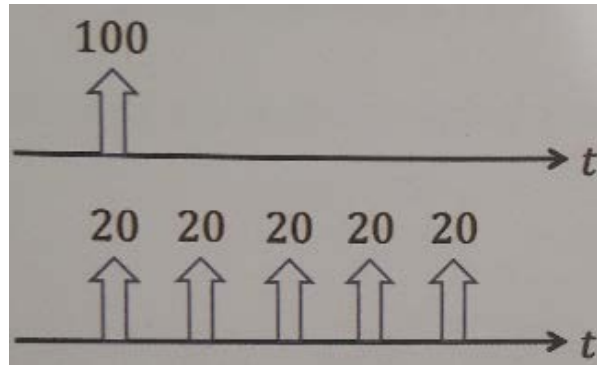
- 보상은 시간마다 받을 수도 있고, 게임이 끝나고 한 번에 받을 수도 있다.
- 보상은 확률변수로 정해져 있는 수가 아니다.
- 보상을 감가하지 않고 더하면 세 가지 문제가 생긴다.

가치함수

- 보상을 단순히 더하였을 때 생기는 문제
 - 에이전트 입장에서 현재의 보상과 미래의 보상이 같아지므로 구분을 할 수 없다.
감가하지 않으면 에이전트가 보는 보상의 합은 단순한 덧셈이기 때문이다.
 - 한 번에 받는 보상과 여러 번 나누어 받는 보상을 구분할 수 없다.
 - 시간이 무한대인 경우 보상이 달라도 그 합은 무한대가 되어 수치적으로 구분할 수 없다.



1. 현재와 미래의 보상



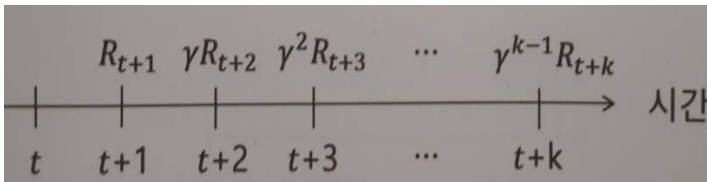
2. 한 번에 받는, 나누어 받는 보상

$$\begin{aligned} 0.1 + 0.1 + 0.1 + \dots &= \infty \\ 1 + 1 + 1 + \dots &= \infty \end{aligned}$$

3. 시간이 무한대일 경우의 보상

가치함수

- 에이전트는 단순한 보상의 합으로 판단을 내리기 어려우므로 정확한 판단을 위해 **감가율을 고려**해야 한다.
- **감가율을 적용한 보상들의 합을 반환값 G_t** 라고 한다.
- 반환값이란 에이전트가 실제로 환경을 탐험하며 받은 보상의 합이고, 받은 보상을 정산하는 것이다.



* 감가율을 고려한 미래 보상의 현재 가치

$$G_t = \underline{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots}$$

* 반환값의 정의

감가율을 적용한
보상들의 합

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_4 + \gamma^3 R_5 + \gamma^4 R_6$$

$$G_2 = R_3 + \gamma R_4 + \gamma^2 R_5 + \gamma^3 R_6$$

$$G_3 = R_4 + \gamma R_5 + \gamma^2 R_6$$

$$G_4 = R_5 + \gamma R_6$$

$$G_5 = R_6$$

* 받은 보상의 정산 : 반환값

에피소드를 t=1~5 진행했다면
에피소드가 끝난 후에
방문했던 상태들에 대한
5개의 반환값이 생긴다.

가치함수

- 에이전트는 반환값을 에피소드가 끝난 후에 알 수 있다.
- 하지만 현재의 정보를 바탕으로 **보상을 예측**하여 행동할 수 있다.
- 어떠한 상태에 있으면 앞으로 얼마의 보상을 받을 것인지에 대한 **기댓값을 고려**하는 것이 "가치함수"이다.
- 기댓값은 반환값의 기댓값으로 표현된다.

$$v(s) = E[G_t | S_t = s]$$

* 가치함수

- 반환값은 보상이 모두 확률적이고 그 보상의 합이므로 확률변수이다.
- 가치함수는 특정 양을 나타내는 값으로 확률변수가 아니다. 따라서 소문자로 표현한다.
- 가치함수는 **에이전트가 가지고 있는 값**이다.
- 상태의 가치를 고려하는 이유는 현재 에이전트가 갈 수 있는 상태의 가치를 알면 그중 가장 가치가 높은 상태를 선택할 수 있기 때문이다.

가치함수의 표현

$$v(s) = E[\underline{G}_t | S_t = s]$$

← 가치함수

$$v(s) = E[\underline{R}_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

← 반환값의 수식을
대입한 가치함수

$$v(s) = E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s]$$

$$v(s) = E[R_{t+1} + \gamma \underline{G}_{t+1} | S_t = s]$$

앞으로 받을 것이라
예상하는 보상
(가치함수로 표현 가능)

← 반환값으로 나타
내는 가치함수

$$v(s) = E[R_{t+1} + \gamma \underline{v}(S_{t+1}) | S_t = s]$$

← 가치함수로 표현
하는 가치함수

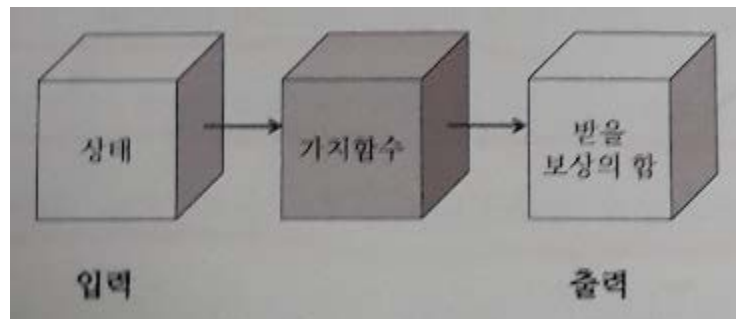
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

← 정책을 고려한
가치함수의 표현

- 정책을 고려한 가치함수의 표현은 벨만 기대 방정식이다.
- 벨만 기대 방정식은 현재 상태의 가치함수($v_\pi(s)$)와 다음 상태의 가치함수($v_\pi(S_{t+1})$) 사이의 관계를 말해주는 방정식이다.
- 강화학습은 벨만 방정식을 어떻게 풀어가느냐의 스토리이다.

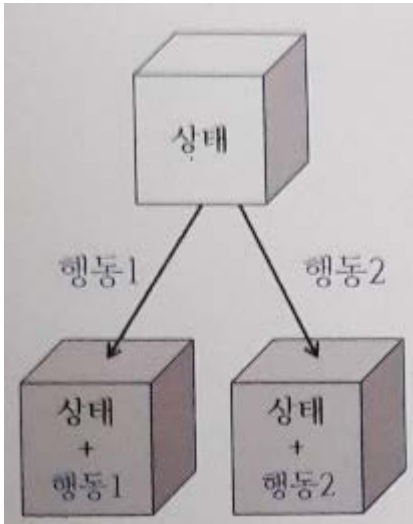
큐함수

- 가치함수는 상태가 입력, 앞으로 받을 보상의 합을 출력으로 하는 "상태 가치함수"이다.
- 에이전트는 가치함수를 통해 어떤 상태로 가야 할지 판단한다.
- 어떤 상태에서 각 행동에 대해 따로 가치함수를 만들어서 그 정보를 얻어올 수 있다면 다음 상태의 가치함수를 따져보지 않아도 어떤 행동을 해야 할지 선택할 수 있다.
- 행동 가치 함수는 어떤 상태에서 어떤 행동이 얼마나 좋은지 알려주는 함수이다.
- 행동 가치함수를 큐함수라고 한다.



* 가치함수

큐함수



- 왼쪽의 그림에서 흰색 상자는 상태, 회색 상자는 특정 행동을 한 상태를 의미한다.
- 행동이 행동1, 행동2로 두 개가 있으면 하나의 상태에서 2개의 행동 상태를 가지는 것이다.
- 2개의 행동 상태에서 따로 가치함수를 계산할 수 있다.(큐함수)
- 큐함수는 상태, 행동이라는 두 가지 변수를 가지고 $q_{\pi}(s, a)$ 라고 나타낸다.

큐함수

1. 각 행동을 했을 때 앞으로 받을 보상인 큐함수를 정책에 곱한다.
2. 모든 행동에 대해 큐함수와 정책을 곱한 값을 더하면 가치함수가 된다.

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s) q_{\pi}(s, a)$$

* 가치함수와 큐함수 사이의 관계식

- 큐함수는 강화학습에서 중요한 역할을 한다.
- 강화학습에서 에이전트가 행동을 선택하는 기준으로 가치함수보다는 보통 큐함수를 사용한다.
- 큐함수를 벨만 기대 방정식의 형태로 나타내면 아래의 수식이 된다.
가치함수의 식과의 차이점은 조건문에 행동이 더 들어간다.

$$q_s(s, a) = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

* 큐함수의 정의

벨만 기대 방정식

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

* 벨만 기대 방정식

- 정책을 반영한 가치함수.
- 벨만 기대 방정식이라고 하는 이유는 식에 기댓값이 들어가기 때문이다.
- 현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계를 식으로 나타낸 것이다.
- 벨만 방정식은 강화학습에서 중요한 부분을 차지한다.

벨만 기대 방정식

- 많은 컴퓨터는 방정식을 식 하나로 푸는 방법보다 계속 계산을 하면서 푸는 방법을 사용한다.

$$1 + 1 + 1 + \dots + 1 = 100$$

* 식 하나로 1을 100번 더하기

$$X=0$$

For i in range(100):

$$X=X+1$$

* 루프를 도는 방법

- 벨만 방정식은 값을 변수에 저장하고 루프를 도는 계산으로 참 값을 알아간다. (다이내믹 프로그래밍)

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

* 벨만 기대 방정식

- $v_{\pi}(s)$ 값을 $E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$ 로 대체한다. 즉, 현재 가치함수 값을 업데이트하는 것이다.

벨만 기대 방정식

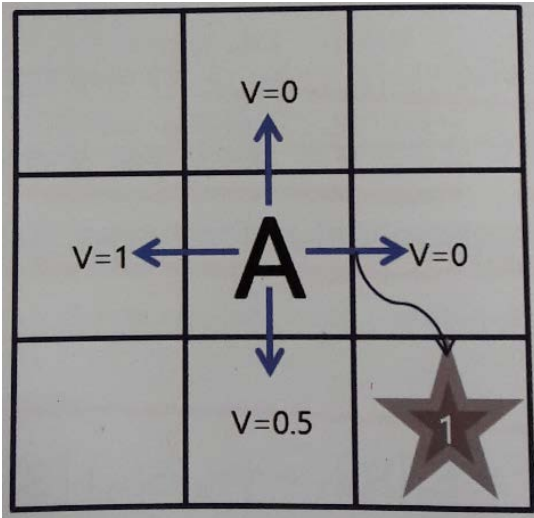
- 현재 가치함수 값을 업데이트하려면 기댓값을 계산해야 한다.

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s) \left(R_{t+1} + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \right)$$

* 계산 가능한 벨만 방정식

- 기댓값에는 정책과 상태 변환 확률이 포함된다.
- 위의 수식은 기댓값의 계산이 가능한 형태로 벨만 기대방정식을 나타낸 것이다.

벨만 기대 방정식



- 상태 변환 확률을 모든 s 와 a 에 대해 1이라 가정한다.
- 정책 = 무작위로 행동(각 행동은 25%의 확률)
- 에이전트의 상태(현재 상태)에 저장된 가치함수 = 0
- 왼쪽 상태의 가치함수 = 1
- 밑의 상태의 가치함수 = 0.5
- 위의 상태의 가치함수 = 0
- 오른쪽 상태의 가치함수 = 0
- 오른쪽 행동을 취할 경우 보상 1
- 감가율 0.9

* 그리드월드에서 가치함수의 업데이트

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s)(R_{t+1} + \gamma v_{\pi}(s'))$$

* 상태 변환 확률이 1인 벨만 기대 방정식

* 위 식의 의미

1. 각 행동에 대해 그 행동을 할 확률을 고려
2. 각 행동을 했을 때 받을 보상 고려
3. 다음 상태의 가치함수 고려

벨만 기대 방정식

1	행동 = 상	$0.25 * (0 + 0.9 * 0) = 0$
2	행동 = 하	$0.25 * (0 + 0.9 * 0.5) = 0.1125$
3	행동 = 좌	$0.25 * (0 + 0.9 * 1) = 0.225$
4	행동 = 우	$0.25 * (1 + 0.9 * 0) = 0.25$
5	기댓값 =	$0 + 0.1125 + 0.225 + 0.25 = 0.5875$

* 벨만 기대 방정식의 계산

- 벨만 기대 방정식을 이용해 현재의 가치함수를 계속 업데이트하다 보면 참값을 구할 수 있다.
- 참값은 최대를 받을 보상이 아니다.
- 참값이란 “현재의 정책을 따라갔을 경우에 에이전트가 얻을 실제 보상의 값에 대한 기댓값”이다.

벨만 최적 방정식

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

* 벨만 기대 방정식

- 처음 가치함수의 값들은 의미 없는 값으로 초기화된다.
- 벨만 기대 방정식을 통해 계속 계산을 무한히 반복하면 식의 왼쪽 항과 오른쪽 항이 동일해진다. ($v_{\pi}(s)$ 수렴, 정책 π 에 대한 참 가치 함수를 구하는 것)
- 참 가치함수란 "어떤 정책을 따라서 움직였을 경우에 받게 되는 보상에 대한 참값"이다.
- 최적 가치함수는 "수많은 정책 중에서 가장 높은 보상을 주는 가치 함수"이다.

벨만 최적 방정식

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s) (R_s^a + \gamma v_k(s'))$$

* 기댓값이 계산 가능한 형태의 벨만 기대 방정식

- $v_{k+1}(s)$ 의 아래 첨자 $k+1$ 은 현재 정책에 따라 $k+1$ 번째 계산한 가치 함수를 뜻하고 상태 s 의 가치함수를 의미한다.
- $k+1$ 번째 가치함수는 k 번째 가치함수 중에서 주변상태들 s' 을 이용해 구한다.
- 이 계산은 상태집합에 속한 모든 상태에 대해 동시에 진행한다. (그리드 월드에선 25개의 상태에 대해 동시에 계산)
- 주변 상태에 저장돼 있는 가치함수를 통해 현재 가치함수를 업데이트 한다.
- 참 가치함수를 구할 수 있다.

벨만 최적 방정식

- 강화학습은 MDP로 정의되는 문제에서 최적 정책을 찾는 것이다.
- 최적 정책은 모든 정책에 대해 가장 큰 가치함수를 주는 정책이다.
- 최적 정책을 따라갔을 때 받을 보상의 합이 최적 가치함수이다.
- max는 정책에 따른 값 중 최대를 반환하는 함수.

$$v^*(s) = \max_{\pi} [v_{\pi}(s)]$$

* 최적의 가치함수

$$q^*(s, a) = \max_{\pi} [q_{\pi}(s, a)]$$

* 최적의 큐함수

- 선택 상황에서 판단 기준은 큐함수이고, 최적 정책은 이 큐함수 중에서 가장 높은 행동 하나를 하는 것이다.
- 최적 큐함수 q^* 만 알면 최적 정책을 알 수 있다.
- argmax는 q^* 를 최대로 해주는 행동 a 를 반환하는 함수.

$$\pi^*(s, a) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a \in A} q^*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

* 최적 정책

벨만 최적 방정식

- 최적 가치함수(or 큐함수)를 구하는 것이 순차적 행동 결정 문제를 푸는 것이다. (구하는 방법은 3장에서)
- 벨만 방정식은 현재상태의 가치함수와 다음 타임스텝 상태의 가치함수 사이의 관계식이다.
- 현재 상태의 가치함수가 최적이라면 에이전트가 가장 좋은 행동을 선택한다.
- 에이전트는 큐함수를 기준으로 가장 좋은 행동을 선택한다. 즉, 큐함수 중 최적의 큐함수를 선택해야 한다.
- 최적의 큐함수 중에서 max를 취하는 것이 최적의 가치함수가 된다.

$$v^*(s) = \max_a [q^*(s, a) | s_t = s, a_t = a]$$

* 큐함수 중 최대를 선택하는 최적 가치함수

벨만 최적 방정식

$$v^*(s) = \max_a [q^*(s, a) | s_t = s, a_t = a]$$

* 큐함수 중 최대를 선택하는 최적 가치함수

큐함수를 가치함수로
고쳐서 표현

$$v^*(s) = \max_a E[R_{t+1} + \gamma v^*(S_{t+1}) | s_t = s, a_t = a]$$

* 벨만 최적 방정식

$$q^*(s, a) = E[R_{t+1} + \gamma \max_{a'} q^*(S_{t+1}, a') | s_t = s, a_t = a]$$

* 큐함수에 대한 벨만 최적 방정식

- 최적 정책을 따라갈 때 **현재 상태의 큐함수**는 **다음 상태에 선택 가능한 행동 중에서 가장 높은 값의 큐함수를 1번 감가하고 보상을 더한 것과 같다.**
- **E**가 붙는 이유는 큐함수 자체가 행동까지 선택한 상황이라 그에 따라 받는 보상은 에이전트가 선택하는 것이 아니라 **환경이 주는 값이기 때문이다.**
- 벨만 기대 방정식과 벨만 최적 방정식을 이용해 MDP로 정의되는 문제를 **계산으로 푸는 방법이 다이내믹 프로그래밍** 이다.

감사합니다