3장 벨만 방정식_문법_정리_버전

CHAPTER 3 벨만 방정식

- 지금까지 다룬 내용은 상태 전이가 결정적이고, 에이전트의 행동도 결정적이었다.
- 그러면 이번에는 상태 전이가 확률적으로 동작할 때 MDP는 어떻게 구할 수 있는지 알아보는 것을 목표로 한다.

---

0. 기초 확률

※ 확률이 익숙한 경우 넘어가도 된다.

주사위

- 확률 변수 x : 주사위 눈의 수 ($x = 1, 2, 3, 4, 5, 6$)
- 각 눈이 나올 확률 $p(x)$: $\frac{1}{6}$
- 주사위 눈의 기댓값 $E[x]$:

$$E[x] = 1 * \frac{1}{6} + 2 * \frac{1}{6} + 3 * \frac{1}{6} + 4 * \frac{1}{6} + 5 * \frac{1}{6} + 6 * \frac{1}{6} = 3.5$$

  현재 주사위의 보상 기댓값은 3.5이다.

주사위 + 동전

진행 조건

- 주사위를 먼저 던진 후, 이어서 동전을 던지는 방식으로 진행한다.
- 주사위가 **짝수**인 경우 : 앞면 0.8 / 뒷면 0.2
- 주사위가 **홀수**인 경우 : 앞면 0.5 / 뒷면 0.5
- 위 확률에 따라 동전이 앞면이 나온 경우 주사위 눈의 수만큼 보상 획득
- 위 확률에 따라 동전이 뒷면이 나온 경우 보상은 0

계산 방법

예시

- 주사위 눈이 1 나올 확률 : $\frac{1}{6}$
- 동전 앞면이 나올 확률 : $\frac{1}{2}$
  -> 이때 보상 1을 얻는다.

- 동전 뒷면이 나올 확률: $\frac{1}{2}$
  -> 이때 보상 0을 얻는다.
  => 주사위 1의 총 얻는 보상은 $\frac{1}{6} * \frac{1}{2} * 1 + \frac{1}{6} * \frac{1}{2} * 0 = \frac{1}{12}$
- 위와 같은 방법으로 각 경우의 수 만큼 계산을 하면 p84의 계산과 같다.

수식으로 표현

- 확률 변수 x : 주사위 눈
- 확률 변수 y : 동전 던지기 결과

$$p(x, y) = p(x)p(y|x)$$

- 이 때, 보상 기댓값은 주사위 눈이 x이고, 동전 던지기 결과가 y가 나왔을 때, 보상값이다.

$$\begin{aligned} E[r(x, y)] &= \Sigma_x \Sigma_y p(x, y) r(x, y) \\ &= \Sigma_x \Sigma_y p(x) p(y|x) r(x, y) \end{aligned}$$

---

1. 벨만 방정식

- **벨만 방정식**이란, 상태 s의 상태 가치 함수와 다음에 취할 수 있는 상태 $s\grave{}$의 상태 가치 함수의 관계를 나타낸 식이다.

벨만 방정식 유도

[시간에 따른 수익]

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} ... = R_t + \gamma G_{t+1}$$

[상태 가치 함수]

$$v_\pi = E_\pi[G_t | S_t = s]$$

위 상태 가치 함수에서 시간에 따른 수익 식을 대입하여 정리하면 다음과 같다.

$$v_\pi = E_\pi[R_t | S_t = s] + \gamma E_\pi[G_{t+1} | s_t = s]$$

여기서 각각의 항을 다음으로 표현할 수 있다.

$$E_\pi[R_t | S_t = s] = \Sigma_{a, s\grave{}} \pi(a|s) p(s\grave{} | s, a) r(s, a, s\grave{})$$

- 상태가 s일 때의 즉시 보상 기댓값은, 정책 π에 따라 행동 a를 선택하고 전이확률 $p(s' \mid s, a)$ 로 도달하는 각 다음 상태 s'에서 얻는 보상 r(s,a,s')의 가중합이다.

$$E_\pi[G_{t+1} | S_t = s] = \Sigma_{a, s\grave{}} \pi(a|s) p(s\grave{} | s, a) v_\pi(s\grave{})$$

- 상태가 s일 때의 다음 수익 기댓값 $E_\pi[G_{t+1} \mid S_t = s]$은 정책 $\pi(a|s)$와 전이 확률 $p(s\grave{}|s, a)$로 가중한 다음 상태 $s\grave{}$ 의 가치 $v_\pi(s\grave{})$의 가중합이다.

위 내용을 정리하여 벨만 방정식을 표현하면 다음과 같다.

$$v_\pi = E_\pi[R_t|S_t = s] + \gamma E_\pi[G_{t+1}|s_t = s] = \Sigma_{a,s`}\pi(a|s)p(s`|s,a)\{r(s,a,s`) + \gamma v_\pi(s`)\}$$

벨만 방정식 예시 문제

두 칸짜리 그리드 월드가 있다고 하고 정책은 다음을 따른다고 하자.

- 50%확률로 오른쪽 또는 왼쪽으로 이동한다.
- 벽에 부딪히면 -1,L2의 사과를 먹으면 1, 그 외는 0의 보상을 받는다.
- 첫번째 칸을 L1, 두번째 칸을 L2라고 정의한다.
- 상태는 결정적으로 정의
- γ는 0.9

위 경우에 대한 벨만 방정식 정의를 하면 다음과 같다.

상태 전이가 결정적(deterministic)으로 이루어지기 때문에 전이 확률 $p(s` \mid s, a)$가 아니라 함수 $f(s,a)$ 값에 따라 정해진다.

$$p(s' \mid s,a) = \begin{cases} 1 & \text{if } s' = f(s,a), \\ 0 & \text{otherwise.} \end{cases}$$

위 정의에 의해 벨만 방정식은 다음과 같다.

$$v_\pi(s) = \Sigma_a \pi(a \mid s)\{r(s,a,s\backslash) + \gamma v_\pi(s\backslash)\}$$

그리고 주어진 식으로 $v_\pi(L1)$와$v_\pi(L2)$ 를 구하면 다음과 같다.

$$\begin{aligned} v_\pi(L1) &= 0.5\,(\text{L1에서 왼쪽으로 가는 경우}) + 0.5\,(\text{L1에서 오른쪽으로 가는 경우}) \\ &= 0.5\{-1 + \gamma v_\pi(L1)\} + 0.5\{1 + \gamma v_\pi(L2)\} \\ &= 0.5\{-1 + 0.9 v_\pi(L1)\} + 0.5\{1 + 0.9 v_\pi(L2)\} \end{aligned}$$

$$\begin{aligned} v_\pi(L2) &= 0.5\,(\text{L2에서 왼쪽으로 가는 경우}) + 0.5\,(\text{L2에서 오른쪽으로 가는 경우}) \\ &= 0.5\{0 + \gamma v_\pi(L1)\} + 0.5\{-1 + \gamma v_\pi(L2)\} \\ &= 0.5\{0 + 0.9 v_\pi(L1)\} + 0.5\{-1 + 0.9 v_\pi(L2)\} \end{aligned}$$

위 두 식을 정리하면 다음과 같이 연립 방정식이 나오고 $v_\pi(L1)$와$v_\pi(L2)$를 구할 수 있다.

$$\begin{cases} -0.55 v_\pi(L1) + 0.45 v_\pi(L2) = 0 \\ 0.45 v_\pi(L1) - 0.55 v_\pi(L2) = 0.5 \end{cases}$$

$$\begin{cases} v_\pi(L1) = -2.25 \\ v_\pi(L2) = -2.75 \end{cases}$$

=> 무작위 정책에서는 L1에서 시작하면 -2.25의 수익을, L2에서 시작하면 -2.75의 수익을 기대할 수 있다.

벨만 방정식에서 행동 가치 함수

- 상태 가치 함수에서 **행동 a**를 조건으로 추가하면 행동 가치 함수 (Q 함수)이다.

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

> 여기서 행동 가치 함수는 상태 s에서는 정책 π와 무관하게 행동 a를 수행한다. 그리고 그 다음 t+1에서 정책 π를 따른다.

수익 G_t를 G_{t+1}의 관계식으로 정리하면 다음과 같고, 계속 식을 정리하면 그 다음과 같다.

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t \mid S_t = s, A_t = a] \\ &= E_\pi[R_t + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= E_\pi[R_t \mid S_t = s, A_t = a] + \gamma E_\pi[G_{t+1} \mid S_t = s, A_t = a] \\ &= \Sigma_{s`} p(s` \mid s,a) r(s,a,s`) + \gamma \Sigma_{s`} p(s` \mid s,a) E_\pi[G_{t+1} \mid S_{t+1} = s`] \\ &= \Sigma_{s`} p(s`| s,a)\{r(s,a,s`) + \gamma e_\pi[G_{t+1} \mid S_{t+1} = s`]\} \\ &= \Sigma_{s`} p(s` \mid s,a)\{r(s,a,s`) + \gamma v_\pi(s`)\} \end{aligned}$$

※ 3번째 줄에서 다음 줄로 전개되는 상세한 식은 다음과 같다.

다음 상태 S_{t+1}가 가질 수 있는 모든 s′에 대해 평균을 내면:

a) 전체 기댓값의 법칙 적용

$$\mathbb{E}_\pi[G_{t+1} \mid S_t = s, A_t = a] = \sum_{s'} p(s' \mid s,a)\, \mathbb{E}_\pi[G_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s']$$

- 다음 상태 S_{t+1}가 여러 후보 s'로 갈 수 있으니, 그 확률로 가중 평균으로 정리한다.

b) 마르코프 성질 적용

$$\mathbb{E}_\pi[G_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'] = \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']$$

- 미래 G_{t+1}는 과거 (s,a)와는 무관하고, 오직 현재 상태 S_{t+1}에만 의존 한다.

---

2. 벨만 최적 방정식

벨만 방정식은 **특정 정책**을 따를 때 각 상태의 가치를 나타내는 식이다.
이 중에서 **모든 상태에서 가치가 최대가 되도록 하는 정책**을 **최적 정책**이라고 부른다.
이 최적 정책에 대해 성립하는 식을 **벨만 최적 방정식**이라고 한다.

벨만 최적 방정식

1절 벨만 방정식에서 상태 가치 함수와 행동 가치 함수에 대해서 알아봤던 것처럼 벨만 최적 방정식에서의 상태 가치 함수와 행동 가치 함수를 정의하면 다음과 같다.

상태 가치 함수

- 벨만 방정식에서 정책 함수를 최적 정책 함수 $\pi_*(a \mid s)$로 바꾸면 벨만 최적 방정식이 성립한다.

$$v_*(s) = \Sigma_a \pi_*(a \mid s) \Sigma_{s`} p(s` \mid s,a)\{r(s,a,s`) + \gamma v_*(s`)\}$$

여기서 최적 정책이란. 정책에 따라 나올 수 있는 행동 값들 중 가장 큰 값을 반환하는 액션을 취하는 정책을 의미한다.
그래서 위 식을 max 함수를 통해 정리하면 다음과 같다.

$$v_*(s) = \max_a \Sigma_{s`} p(s` \mid s, a)\{r(s, a, s`) + \gamma v_*(s`)\}$$

행동 가치 함수

- 벨만 방정식에서 정책 함수를 최적 정책 함수 π_*로 바꾸면 벨만 최적 방정식이 성립한다.

$$\begin{aligned} q_\pi(s, a) &= \Sigma_{s`} p(s` \mid s, a)\{r(s, a, s`) + \gamma v_*(s`)\} \\ &= \Sigma_{s`} p(s` \mid s, a)\{r(s, a, s`) + \gamma \Sigma_{a`} \pi_*(a` \mid s`) q_*(s` \mid a`)\} \\ &= \Sigma_{s`} p(s` \mid s, a)\{r(s, a, s`) + \gamma \max_{a`} q_*(s` \mid a`)\} \end{aligned}$$

벨만 최적 방정식 예시 문제

두 칸짜리 그리드 월드가 있다고 하고 정책은 다음을 따른다고 하자.

- 벽에 부딪히면 -1,L2의 사과를 먹으면 1, 그 외는 0의 보상을 받는다.
- 첫번째 칸을 L1, 두번째 칸을 L2라고 정의한다.
- 상태는 결정적으로 정의
- γ는 0.9

상태 전이가 결정적(deterministic)으로 이루어지기 때문에 전이 확률 $p(s` \mid s, a)$가 아니라 함수 $f(s, a)$ 중 가장 큰 값에 따라 결정된다.

위 경우에 대한 벨만 최적 방정식 정의를 하면 다음과 같다.

$$v_*(s) = \max_a \{r(s, a, s`) + \gamma v_*(s`)\}$$

그리고 주어진 식으로 $v_\pi(L1)$와$v_\pi(L2)$ 를 구하면 다음과 같다.

$$v_*(L1) = \max \begin{cases} -1 + 0.9 v_*(L1), \\ 1 + 0.9 v_*(L2) & \text{(가장 큰 경우의 수)} \end{cases}$$

$$v_*(L2) = \max \begin{cases} 0.9 v_*(L1), & \text{(가장 큰 경우의 수)} \\ -1 + 0.9 v_*(L2) \end{cases}$$

총 4가지의 경우의 수를 정의하여 직접 계산한 뒤, 가장 큰 값을 구하면 다음과 같다.

$$v_*(L1) = 5.26, v_*(L2) = 4.73$$

※ 여기서 잠깐 ✋ !

> 여기서 사용되는 **max 연산은 비선형 함수**이다.
> 경우의 수가 적을 때는 직접 어떤 경우가 최대를 주는지 계산할 수 있지만,
> 일반적으로는 선형 방정식 풀이처럼 단순한 계산으로는 해결할 수 없다.
> 이런 경우에는 **비선형 방정식 풀이 기법**을 사용해야 하며,
> 이에 대한 자세한 내용은 마지막에서 따로 다룬다.

최적 정책 구하기

바로 위에서 언급했던 것 처럼 최적 정책을 손으로 풀이하여 풀 수 있으면 가장 좋겠지만, 사실 그럴 수 있는 문제는 별로 없다.
그래서 최적 정책 구하는 법을 알아보면 다음과 같다.

최적 행동 가치 함수 $q_*(s, a)$를 알고 있다고 가정하면 상태 s에서의 최적 행동은 다음과 같다.

$$\mu_*(s) = \underset{a}{\operatorname{argmax}}\, q_*(s, a)$$

여기서 최적 행동 가치 함수를 대입하여 정리하면 다음과 같다.

$$\mu_*(s) = \underset{a}{\operatorname{argmax}}\, q_\pi(s, a) = \underset{a}{\operatorname{argmax}}\, \Sigma_{s`} p(s` \mid s, a)\{r(s, a, s`) + \gamma v_*(s`)\}$$

최적 정책 구하기 예시 문제

벨만 최적 방정식 예시 문제를 통해서 얻은 최적 상태 가치 함수를 가져와 보자.

$$v_*(L1) = 5.26, v_*(L2) = 4.73$$

L1에서 왼쪽으로 갔을 경우

$$-1 + 0.9v_*(L1) = -1 + 0.9 * 5.26 = 3.734$$

L1에서 오른쪽으로 갔을 경우

$$1 + 0.9v_*(L2) = 1 + 0.9 * 4.73 = 5.257$$

즉 L1에서는 **오른쪽**으로 움직이는게 최적 행동이고, 같은 방식으로 L2도 해보면 **왼쪽**으로 움직이는게 최적 행동으로 나온다.

추가 정리 사항

- **선형 방정식 계산기**: 연립 방정식 풀이처럼, 미지수들을 선형 방정식으로 두고 산술 계산(행렬 연산 등)을 통해 직접 해를 구하는 방법을 의미한다.
- **비선형 방정식 계산기**: 단순한 산술 풀이로는 답을 구할 수 없기 때문에,
초기 값을 설정한 뒤 반복적으로 갱신하며 최적 해에 수렴하는 방법(Value Iteration),
혹은 정책을 가정하고 평가·개선 과정을 반복하며 최적 정책을 찾는 방법(Policy Iteration)을 의미한다.

예시 문제

- 상태: $S = \{A, B\}$
- 행동: $a \in \{1, 2\}$
- 보상과 전이는 다음과 같다고 하자.

상태	행동	보상	다음 상태
A	1	2	B
A	2	0	A
B	1	1	A
B	2	3	B

할인율 $\gamma = 0.9$.

Value Iteration (값 반복)

- "가치 함수 v(s)v(s)v(s)를 0으로 시작해서, 계속 업데이트하면 최적 값에 가까워진다."

초기 값

$$v_0(A) = 0, v_0(B) = 0$$

업데이트 식

$$v_{k+1}(s) = \max_a \{r(s,a) + \gamma v_k(s\prime)\}$$

1회 업데이트

- 상태 A:
  - 행동 1: $2 + 0.9 \cdot v_0(B) = 2 + 0 = 2$
  - 행동 2: $0 + 0.9 \cdot v_0(A) = 0$
    => max=2, 따라서 $v_1(A) = 2$.
- 상태 B:
  - 행동 1: $1 + 0.9 \cdot v_0(A) = 1 + 0 = 1$
  - 행동 2: $3 + 0.9 \cdot v_0(B) = 3 + 0 = 3$
    ⇒ max=3, 따라서 $v_1(B) = 3$.

2회 업데이트

- 상태 A:
  - 행동 1: $2 + 0.9 \cdot v_1(B) = 2 + 0.9 * 3 = 4.7$
  - 행동 2: $0 + 0.9 \cdot v_1(A) = 0.9 * 2 = 1.8$
    => max=4.7, 따라서 $v_2(A) = 4.7$.
- 상태 B:
  - 행동 1: $1 + 0.9 \cdot v_1(A) = 1 + 0.9 * 2 = 2.8$
  - 행동 2: $3 + 0.9 \cdot v_1(B) = 3 + 0.9 * 3 = 5.7$
    ⇒ max=5.7, 따라서 $v_2(B) = 5.7$.
- 👉 이런 식으로 계속 업데이트하면 v(A),v(B)가 수렴하고, 최적 정책은 "항상 max를 준 행동"이

된다.

$$v(A) \approx 29, v(B) \approx 30$$

$$\pi_*(A) = 행동1, \pi_*(B) = 행동2$$

Policy Iteration (정책 반복)

1. 아무 정책 π를 정한다.
2. 그 정책의 가치를 계산한다(정책 평가).
3. 더 나은 행동이 있으면 바꾼다(정책 개선).
4. 바꿀 게 없으면 최적 정책.

1. 초기 정책 가정

- A: 행동 2
- B: 행동 1

2. 정책 평가

현재 정책에 대해 **벨만 방정식**을 풀어 가치 함수 계산한다.

- A에서 행동 2만 한다 : $v(A) = 0 + 0.9v(A) \Rightarrow v(A) = 0$
- B에서 행동 1만 한다 : $v(B) = 1 + 0.9v(A) = 1 + 0 = 1$

3. 정책 개선

이제 각 상태에서 다른 행동을 비교한다.

- A:
  - 정책 행동 2 → 0
  - 다른 행동 1 → $2 + 0.9v(B) = 2 + 0.9 \cdot 1 = 2.9$
    ⇒행동 2보다 행동 1이 더 크므로 A의 정책을 행동 1로 바꿈.
- B:
  - 정책 행동 1 → 1
  - 다른 행동 2 → $3 + 0.9v(B) = 3 + 0.9 \cdot 1 = 3.9$
    ⇒ 행동 1보다 행동 2가 더 크므로 B의 정책을 행동 2로 바꿈.

4. 새로운 정책

$$A \to 1, B \to 2$$

초기 설정한 정책에서 새로운 정책이 정해졌다.
다시 지금까지 과정을 반복하여 새로운 정책을 구하면 최적 정책 $\pi_*(A) = 1, \pi_*(B) = 2$에 도달하게 된다.