

## 1. 순차 행동 결정 문제 푸는 과정

순차적 행동 결정 문제를 푸는 과정은 다음과 같다.



1. 순차적 행동 문제를 MDP로 전환한다
2. 가치함수를 벨만 방정식으로 반복적으로 계산한다
3. 최적 가치함수와 최적 정책을 찾는다

$$\widehat{v}(s) = E[G] \mid S_t = s$$

수식 2.21 가치함수

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

수식 2.25 정책의 정의  
정책을 고려한 가치함수의 표현

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

수식 2.15 정책의 정의  
어떠한 행동을 할 확률

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

수식 2.12 상태 변화 확률  
그 행동을 했을 때 어떤 상태로 가게 될 확률

## 2. 가치함수

가치함수는 현재 상태의 정책을 따라갔을 때 최종적으로 받을 것이라 예상되는 보상의 총 합이다.

에이전트는 가치함수를 통해 보상의 합을 최대로 한다는 목표에 얼마나 다가갔는지를 알 수 있다.  
왜냐하면 가치함수는 에이전트가 받을 보상의 합에 대한 기대값이기 때문이다.

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

수식 2.25 정책을 고려한 가치함수의 표현

(에이전트는 정책을 업데이트 할 때 가치함수보다는 큐함수를 사용한다. 에이전트는 선택할 각 행동의 가치를 직접 나타낼 수 있게 하기 위해서)

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

수식 2.27 큐함수의 정의

이 때 여러 정책들 중 가장 큰 가치함수값을 주는 정책이 최적의 정책이다. 따라서 최적의 정책을 찾기 위해서는 모든 가능한 정책들에 대한 가치함수 값 중에서 최대를 반복하는 가치함수를 찾아야 한다.

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

수식 2.36 최적의 가치함수

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

수식 2.37 최적의 큐함수

정리하면, 모든 정책에 대한 가치함수 중 가장 큰 가치함수값을 반복하는 것이 바로 최적 가치함수이고, 그 때의 정책이 최적 정책이다.

이 가치함수에 대한 방정식이 벨만 방정식이고, 벨만 방정식에는 두 가지 종류가 있다. 벨만 기대 방정식과 벨만 최적 방정식.

## 3. 벨만 방정식

### 벨만 방정식

벨만 방정식의 정의는 다음과 같다. “현재 상태와 다음 상태의 관계를 나타내는 방정식”

• 벨만 기대 방정식:

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

수식 2.35 기댓값의 계산 가능한 형태의 벨만 기대 방정식

반복적으로 기대값을 업데이트해 나가기 위해 현재의 가치함수와 다음의 가치함수 사이의 관계를 정의한 방정식이다.  
벨만 기대 방정식은 특정 정책일 때의 가치함수 사이의 관계다. 이렇게 모든 상태에 대한 가치함수를 반복적으로 업데이트하면 참 가치함수값이 나온다. 즉 현재 정책  $P_t$ 에 대한 참 가치함수값(수렴된 값)이 나온다.

• 벨만 최적 방정식:

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

가장 큰 가치함수를 주게 하는 정책이 최적 정책이라고 할 수 있다. 벨만 기대 방정식에 따라 더 좋은 정책을 계속 찾아나가다 보면 최적의 정책을 찾게 된다.

최적의 가치함수값을 내는 것을 ‘최적 가치함수’라 하는데, 최적 가치함수를 받게 한 그 정책이 바로 최적 정책이다.

이 때의 가치함수를 사이의 관계식을 벨만 최적 방정식이라 한다.

여러 가치함수들 중에서 가장 큰 가치함수를 최적 가치함수로 보고, 그 때의 정책을 최적 정책으로 본다.

다이내믹 프로그래밍에는 정책 이터레이션과 가치 이터레이션이 있다.

정책 평가는, 현재의 정책을 대체로 반복적으로 업데이트하는 것이다.( $k$ 번째 함수를 통해  $k+1$ 번째 함수를 업데이트하는 것)

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

각 상태의 가치함수를 구하고,

구한 가치함수를 업데이트한다음,

업데이트된 가치함수에서 다시 가치함수를 구하고,

다시 업데이트하는 과정을 반복한다.

이 과정을 반복하여 참 가치함수값을 구하게 된다.(결국  $v^*(s)$  값에 수렴하게 됨)

이런 방식이 다이내믹 프로그래밍으로 방정식을 푸는 방식이다.

다이내믹 프로그래밍에는 정책 이터레이션과 가치 이터레이션이 있다.

정책 이터레이션은 벨만 기대 방정식을 이용하여 문제를 푸는 것이다. 정책 평가는, 현재 상태에 대해 행동이 정의되어 있다.

가장 높은 보상을 내는 정책을 얻는 것이 목표일 때,

초기에는 정책이 무작위적이다.



이 정책을 시작으로 점차 발전(업데이트)되거나게 된다.

정책을 발전시키기 위해서는 정책을 평가해야 한다. --> 정책 평가(Policy Evaluation)

그 과정은 다음과 같다.



문제는 더 먼 미래를 고려하면 할 수록 경우의 수가 매우 크게 늘어나게 된다.

이 때 앞에서 본 것처럼 다이나믹 프로그래밍을 이용하여 문제를 작게 쪼개고 작은 문제를 푼 결과를 저장하여, 저장된 값을 다음 계산에 이용해 문제를 푸는 방식이다.

이 과정을 반복하면 정책이 최적 정책으로 수렴하게 됨.

여러 가지 방법으로는 탐욕 정책 평가를 사용하는 것이다. 탐욕 정책 평가는, 현재 상태에 대해 반복적으로 정책을 최적 정책으로 본다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

각 상태에서 어떤 행동을 하는 것이 가장 좋을지를 결정하는 것은 큐함수를 이용하면 된다.

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

수식 2.27 큐함수의 정의

기대값을 컴퓨터가 계산하기 쉬운 형태로 바꾼 수식이다.

$\pi(a \mid s) (R_s^a + \gamma v_{\pi}(s'))$

정책 평가는, 현재 상태  $s$ 에서 갈 수 있는 주변 상태  $s'$ 에 저장된 가치함수  $v_{\pi}(s')$ 를 불러온 다음,

감가율을 곱하고,

주변 상태  $s'$ 으로 가는 행동  $a$ 에 대한 보상을 더한다음,

그 행동  $a$ 를 할 확률을 곱하고,

가능한 모든 행동들에 대해서 수행한 결과를 더한다;

이렇게 해서 구한 값을  $k+1$ 번째 가치함수 행렬의 상태  $s$  자리에 저장한다.

이 때 앞에서 본 것처럼 다이나믹 프로그래밍을 이용하여 문제를 작게 쪼개고 작은 문제를 푸는 정책 평가를 통해  $k+1$ 번째 가치함수를 계산하기 위해, 현재 상태인  $k$ 번째 가치함수를 고려하는 것을 관찰할 수 있다.

이 과정을 반복하면 정책이 최적 정책으로 수렴하게 됨.

## 6-1. 정책 평가

벨만 방정식을 푸는 방법은 가치함수(현재의 정책을 따라갔을 때 받게 될 보상에 대한 기대값)를 이용하는 것이다.

만약 세 개의 상태만 존재하고 있다고 가정할 때, 현재 정책에서의 각 상태의 칠 가치함수를 구하는 과정은 다음과 같다. (모든 상태가 동시에 진행된다.) 만약 그리드월드라면 25개의 상태에 대해 모두 동시에 수행된다.

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

수식 2.35 기댓값의 계산 가능한 형태의 벨만 기대 방정식

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

수식 2.41 큐함수에 대한 벨만 최적 방정식

## 4. 다이나믹 프로그래밍

벨만 방정식의 정의는 다음과 같다. “현재 상태와 다음 상태의 관계를 나타내는 방정식”

• 벨만 기대 방정식:

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

수식 2.35 기댓값의 계산 가능한 형태의 벨만 기대 방정식

반복적으로 기대값을 업데이트해 나가기 위해 현재의 가치함수와 다음의 가치함수 사이의 관계를 정의한 방정식이다.

벨만 기대 방정식은 특정 정책일 때의 가치함수 사이의 관계다. 이렇게 모든 상태에 대한 가치함수를 반복적으로 업데이트하면 참 가치함수값이 나온다. 즉 현재 정책  $P_t$ 에 대한 참 가치함수값(수렴된 값)이 나온다.

• 벨만 최적 방정식:

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

가장 큰 가치함수를 주게 하는 정책이 최적 정책이라고 할 수 있다. 벨만 기대 방정식에 따라 더 좋은 정책을 계속 찾아나가다 보면 최적의 정책을 찾게 된다.

이 가치함수에 대한 방정식이 벨만 최적 방정식이고, 벨만 최적 방정식에는 두 가지 종류가 있다. 벨만 기대 방정식과 벨만 최적 방정식.

## 5. 정책 이터레이션

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

각 상태의 가치함수를 구하고,

구한 가치함수를 업데이트한다음,

업데이트된 가치함수에서 다시 가치함수를 구하고,

다시 업데이트하는 과정을 반복한다.

이 과정을 반복하여 참 가치함수값을 구하게 된다.(결국  $v^*(s)$  값에 수렴하게 됨)

이런 방식이 다이나믹 프로그래밍으로 방정식을 푸는 방식이다.

다이나믹 프로그래밍에는 정책 이터레이션과 가치 이터레이션이 있다.

정책 이터레이션은 벨만 기대 방정식을 이용하여 문제를 푸는 것이다. 정책 평가는, 현재 상태에 대해 행동이 정의되어 있다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

정책 평가는, 정책 평가를 통해 평가하고, 그 평가한 결과를 기준으로 정책을 업데이트하는 것이다.

각 상태에서 어떤 행동을 하는 것이 가장 좋을지를 결정하는 것은 큐함수를 이용하면 된다.

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

수식 2.27 큐함수의 정의

기대값을 컴퓨터가 계산하기 쉬운 형태로 바꾼 수식이다.

$\pi(a \mid s) (R_s^a + \gamma v_{\pi}(s'))$

정책 평가는, 현재 상태  $s$ 에서 갈 수 있는 주변 상태  $s'$ 에 저장된 가치함수  $v_{\pi}(s')$ 를 불러온 다음,

감가율을 곱하고,

주변 상태  $s'$ 으로 가는 행동  $a$ 에 대한 보상을 더한다음,

그 행동  $a$ 를 할 확률을 곱하고,

가능한 모든 행동들에 대해서 수행한 결과를 더한다;

이렇게 해서 구한 값을  $k+1$ 번째 가치함수 행렬의 상태  $s$  자리에 저장한다.

이 과정을 반복하면 정책이 최적 정책으로 수렴하게 됨.

## 6-2. 정책 평가 과정

벨만 방정식을 푸는 방법은 가치함수(현재의 정책을 따라갔을 때 받게 될 보상에 대한 기대값)를 이용하는 것이다.

만약 세 개의 상태만 존재하고 있다고 가정할 때, 현재 정책에서의 각 상태의 칠 가치함수를 구하는 과정은 다음과 같다. (모든 상태가 동시에 진행된다.) 만약 그리드월드라면 25개의 상태에 대해 모두 동시에 수행된다.

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

수식 2.35 기댓값의 계산 가능한 형태의 벨만 기대 방정식

기대값을 컴퓨터가 계산하기 쉬운 형태로 바꾼 수식이다.