Chapter 2 마르코프 결정 과정

밴디트 문제에서는 에이전트가 어떤 행동을 취하든 다음에 도전할 문제의 설정은 변하지 않았다. 슬롯머신을 여러 개 돌려도 슬롯머신의 승률으 고정되었다.

하지만, 현실의 문제들은 매번 상황이 바뀌게 된다. 바둑의 예시로 에이전트가 어떤 수를 두면 바둑판 위의 돌 배치가 달라지고, 상대가 돌을 두면 또 달라진다. 이렇게 에이전트의 행동에 따라 상황이 시시각각 변하게 된다. 그러니 에이전트는 상황이 변하는 것을 고려하여 최선의 수를 두어야 한다.

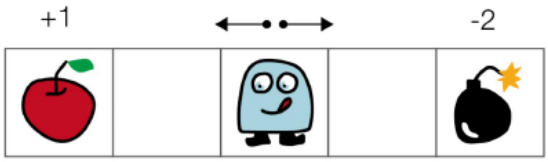
여기서 다룰 문제는 시간에 따라 변화는 상황이 아니라 '에이전트의 행동에 따라' 환경의 상태가 변하는 문제를 다룬다.

2.1. 마르코프 결정 과정(MDP)이란?

마르코프 결정 과정에서 '결정 과정'이란 에이전트가 행동을 결정하는 과정을 뜻한다.

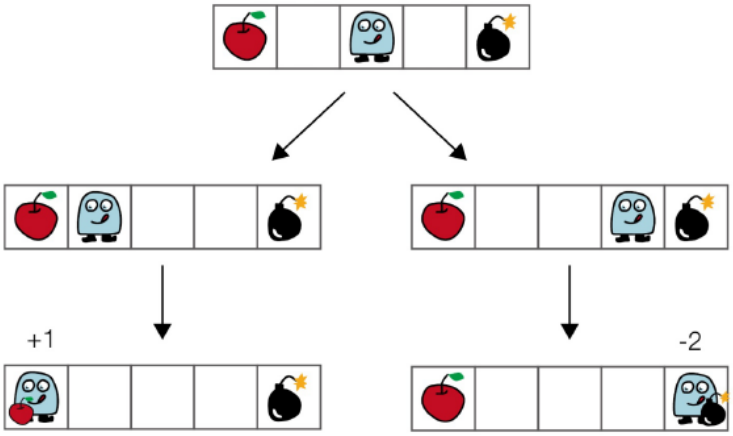
2.1.1. 구체적인 예

세상은 격자로 구분되어 있고, 그 안에 로봇이 있다. 로봇은 격자 세상에서 오른쪽 또는 왼쪽으로 이동할 수 있다.



용어로 얘기하면 로봇이 에이전트이고 주변이 환경이다. 에이전트는 오른쪽으로 이동하거나 왼쪽으로 이동하는 두 가지 행동을 취할 수 있다. 또한, 가장 왼쪽 칸의 사과와 가장 오른쪽 칸의 폭탄은 로봇에게 주어지는 보상이다. 사과를 얻을 때는 +1, 폭탄을 얻을 때는 -2, 빈칸의 보상은 0 이다.

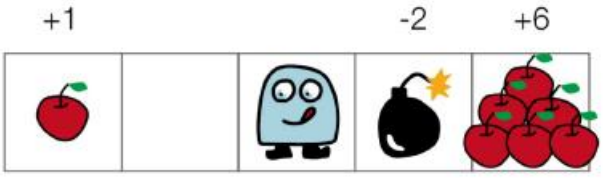
이 문제에서는 에이전트의 행동에 따라 에이전트가 처하는 상황이 달라진다.



이 상황을 **상태**라고 한다. MDP에서는 에이전트의 행동에 따라 상태가 바뀌고, 상태가 바뀐 곳에서 새로운 행동을 하게 된다. 그림에서 보면 에이전트에게 최선의 행동은 왼쪽으로 두 번 이동하는 것이다.

cf) 이해하기: 왜 환경은 그대로인데 상태가 변한다고 할까?

환경 자체는 그대로 지만, 로봇이 경험하는 정보가 달라진다. 로봇이 현재 어디에 있느냐에 따라 앞으로의 선택이 다르게 보일 수 있다. 즉, 같은 행동을 해도 얻는 정보가 다를 수 있다. (로봇이 왼쪽으로 이동한 후, 그 위치에서 왼쪽/오른쪽이 다른 결과를 초래할 수 있다.)
상태가 변하면 행동의 가치도 변하게 된다. 로봇이 현재 있는 위치에 따라 앞으로의 행동이 최적인지 아닌지 달라질 수 있다. 같이 행동이 주는 보상의 기댓값이 로봇의 위치(상태)에 따라 달라지게 된다.
그래서 상태가 변한다는 것은 단순히 물리적인 변화가 아니라, 에이전트가 경험하는 정보가 변화하는 것까지 모두 포함하는 개념이다.

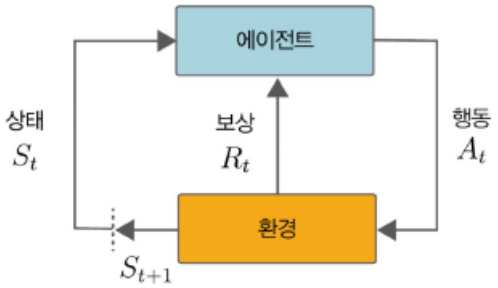


이 그림에서는 오른쪽 끝에 +6짜리 보상이 있다. 오른쪽으로 이동하면 즉시 얻는 보상은 마이너스지만, 한번 더 이동하면 최적의 보상을 얻을 수 있기 때문에, 이 문제에서 최선의 행동은 오른쪽으로 두 번 이동하는 것이다.

따라서, 에이전트는 눈앞의 보상이 아니라 미래에 얻을 수 있는 보상의 총합을 고려해야 한다. 즉, 보상의 총합을 극대화하려 노력해야 한다.

2.1.2. 에이전트와 환경의 상호작용

MDP에서는 에이전트와 환경 사이에 상호작용이 이루어진다. 여기서 에이전트가 행동을 취함으로써 상태가 변하고, 그에 따라 얻을 수 있는 보상도 달라진다는 것이다.



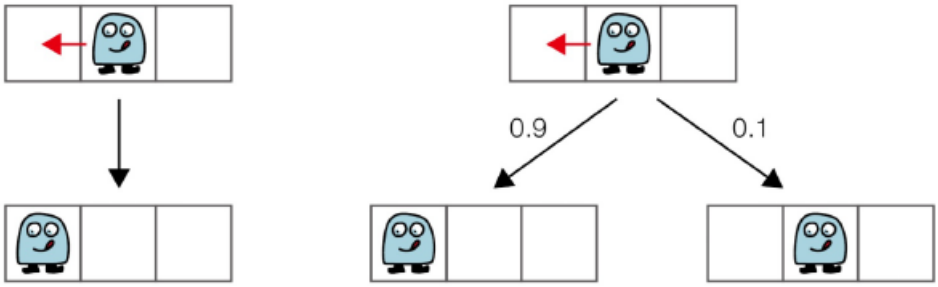
시간 t에서의 상태가 S_t이다. 이 상태에서 시작하여 에이전트가 행동 A_t를 수행하여 보상 R_t를 얻고, 다음 상태인 S_{t+1}로 전환된다. 계속해서 상태가 주어지고, 이에 대해 행동을 하고 보상을 얻고 한 단위 만큼 상태가 변하게 된다.

2.2. 환경과 에이전트를 수식으로

MDP는 에이전트와 환경의 상호작용을 수식으로 표현한다. 그렇게 하려면 다음의 세 요소를 수식으로 표현해야 한다.

- 상태 전이: 상태는 어떻게 전이되는가?
- 보상: 보상은 어떻게 주어지는가?
- 정책: 에이전트는 행동을 어떻게 결정하는가?

2.2.1. 상태전이



왼쪽 그림에서는 에이전트가 왼쪽으로 이동하는 행동을 선택했고, 그 결과로 에이전트가 '반드시' 왼쪽으로 이동한다. 이러한 성질을 결정적이라고 한다. 상태 전이가 결정적이면, 다음 상태 s'는 현재 상태 s와 행동 a에 의해 '단 하나로' 결정된다. 함수로는 다음과 같이 표현할 수 있다.

$$s' = f(s,a)$$

$f(s,a)$는 상태 s와 a를 입력하면 다음 상태 s'를 출력하는 함수이다. 이 함수를 **상태 전이 함수**라고 한다.

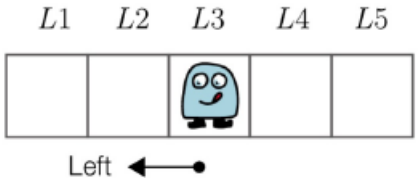
반면, 오른쪽 그림은 이동이 확률적으로 된다. 에이전트가 왼쪽으로 이동하는 행동을 선택하더라도 0.9의 확률로만 왼쪽으로 이동하고, 0.1의 확률로는 그 자리에 머물러 있다. 이러한 확률적 상태 전이 상태에서는 에이전트가 상태 s에서 행동 a를 선택할 때, 이동 확률을 다음과 같이 표현할 수 있다.

$$p(s' \mid s,a)$$

기호 |의 오른쪽에는 '조건'을 나타내는 확률 변수를 적는다. 이 문제에서는 '상태 s에서 행동 a를 선택했다'는 것이 조건에 해당한다. 이 수식을 **상태 전이 확률**이라고 한다.

cf)이해하기: 상태 전이가 결정적, 비결정적
상태 전이는 현재 상태에서 특정 행동을 했을 때 다음 상태가 어떻게 변하는지를 나타내는 것. 상태전이가 결정적이라는 것은 같은 행동을 했을 때 항상 동일한 다음 상태로 이동한다는 의미이다.
반면 비결정적은 현재 상태에서 행동 a를 하면 여러 개의 가능성이 있는 다음 상태 s'들이 존재하고, 각각 확률이 있다는 것이다. 예를 들어서 미끄러운 얼음판에서 오른쪽으로 이동하려고 했을 때, 80%의 확률로 오른쪽으로 가지만 20% 확률로 미끄러져서 아래로 이동할 수 있다는 것이다.

현실에서도 같은 행동을 하더라도 항상 같은 결과가 나오지 않는 경우가 많다. (비결정적 상태 전이) 그 이유는 외부 환경이 항상 동일하지 않기 때문일 수 있다. (같은 속도로 운전해도 날씨, 도로 상태에 따라 도착 시간이 다름 / 자율 주행차의 경우 다른 운전자의 돌발 행동으로 예상과 다르게 반응해야 할 수 있음) 또한, 센서의 오류로 다른 방향으로 이동할 수 있고, 강화학습 에이전트가 수행하는 행동이 완벽하지 않을 수도 있는 등 다양한 요인들이 개입하면서, 상태 전이가 완전히 결정적이기 어렵고 대부분 비결정적으로 된다.



s'	$L1$	$L2$	$L3$	$L4$	$L5$
$p(s'\|s,a)$	0	0.9	0.1	0	0

그림과 같이 총 5개의 칸으로 나누고, 에이전트의 행동에 대해서는 Left, Right로 표기한다. 이제 에이전트가 L3에 있을 때, (상태가 L3일 때), 행동을 Left를 선택했다면 상태 전이확률은 오른쪽 표와 같다.

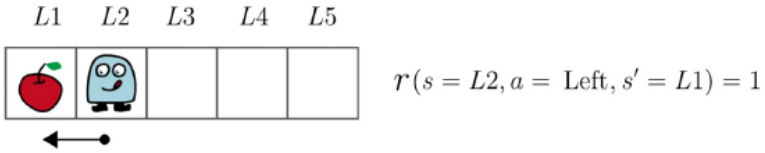
$\rho(s'|s,a)$가 다음 상태 s'를 결정하는 데는 '현재' 상태 s와 행동 a만이 영향을 준다. 이는 과거의 정보, 지금까지 어떤 상태들을 거쳐 왔고 어떤 행동들을 취했는지는 신경 쓰지 않는다. 이처럼 현재 정보만 고려하는 성질을 **마르코프 성질**이라고 한다.

MDP는 마르코프 성질을 만족한다고 가정하고 상태 전이와 보상을 모델링 한다. 만약 마르코프 성질을 따른다고 가정하지 않는다면 과거의 모든 상태와 행동까지 고려해야 해서, 그 조합이 기하급수적으로 많아진다.

cf) 강화학습에서 마르코프 성질을 고려해야 하는 이유?
강화학습에서는 에이전트가 현재 상태에서 행동을 통해 보상을 얻고, 그에 따라 상태가 변하면서 학습을 진행한다. 마르코프 성질을 고려하면, 에이전트가 이전 상태와 행동을 무시하고 현재 상태만을 기반으로 최적의 결정을 내릴 수 있게 된다.
현재 상태만으로도 충분히 최적의 행동을 선택할 수 있기 기 때문에 학습이 효율적으로 진행된다. (자율주행차가 운전하는 도로 상황에서 현재 차선과 속도만 알면 앞으로 무엇을 해야 할지 결정할 수 있다. 과거의 교차로 상태나 얼마나 빨리 왔는지는 중요하지 않게 된다.) 또한, 과거를 고려하지 않아도 되기 때문에 계산 량이 줄어들고, 마르코프 성질을 따르는 환경에서는 최적 정책을 정의하기 쉬어진다.

2.2.2. 보상 함수

보상은 '결정적'으로 주어진다고 가정한다. 에이전트가 상태 s에서 행동 a를 수행하여 다음 상태 s'가 되었을 때 얻는 보상을 r(s,a,s')라는 함수로 정의한다.



그림에서 에이전트가 상태 L2에서 행동 Left를 선택하여 상태 L1로 전이된 예가 있다. 이때 보상 함수를 통해 1임을 알 수 있다. 이 그림에서는 다음 상태 s'만 보면 보상이 결정된다. (이동한 위치에 사과가 있느냐에 의해서만 보상이 결정되기 때문이다.) 그래서 보상 함수를 r(s')라고 할 수도 있다.

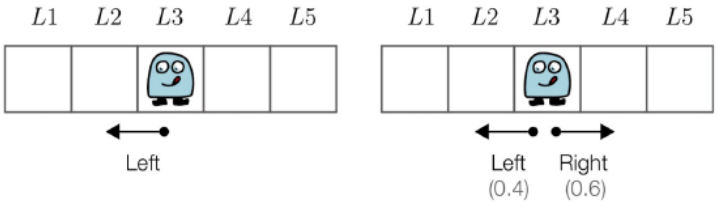
2.2.3. 에이전트의 정책

정책은 에이전트가 행동을 결정하는 방식을 말한다. 정책에서 중요한 점은 에이전트는 '현재 상태'만으로 행동을 결정할 수 있다는 것이다. 그 이유는 환경의 상태 전이가 마르코프 성질에 따라 이루어지기 때문이다.

cf) 마르코프 성질이 현재 상태만으로 최적 행동을 결정할 수 있다고 판단하는 이유
마르코프 성질이란 현재 상태가 주어지면, 미래 상태는 과거 상태에 의존하지 않는다는 성질이다. 즉, 현재 상태가 환경에 대한 완전한 정보를 포함하고 있기 때문에, 과거 정보를 추가로 참조하지 않아도 최적의 행동을 결정할 수 있다는 것이다. (체스 게임에서 과거의 움직임을 따로 기억하지 않아도, 현재 상태만으로 최적의 수를 결정할 수 있다.)

환경의 상태 전이에서는 현재 상태 s와 행동 a만을 고려하여 다음 상태 s'가 결정되고, 보상도 그대로 결정된다. 이는 '환경에 대한 필요한 정보는 모두 현재 상태에 있다'는 것이다. 따라서 에이전트가 '현재 상태'만으로 행동을 결정할 수 있다.

따라서, 에이전트는 현재 상태를 보고 행동을 결정한다. 이때 행동을 결정하는 방식인 정책은 '결정적' 또는 '확률적'인 경우로 나눌 수 있다.



왼쪽 그림은 '반드시' 정해진 행동을 한다. 에이전트는 L3에 있을 때는 반드시 왼쪽으로 이동한다. 이러한 결정 정책은 함수로 다음과 같이 정의할 수 있다.

$$a = \mu(s)$$

μ(s)는 매개변수로 상태를 건내주면 행동 a를 반환하는 함수다.

오른쪽 그림은 확률적 정책의 예이다. 에이전트가 왼쪽으로 이동할 확률은 0.4이고, 오른쪽으로 이동할 확률은 0.6이다. 이렇게 에이전트의 행동이 확률적으로 결정되는 정책은 수식으로 다음과 같이 표현할 수 있다.

$$\pi(a|s)$$

$\pi(a|s)$는 상태 s에서 행동 a를 취할 확률을 나타낸다. 그림의 예를 수식으로 표현하면 다음과 같다.

$$\pi(a = \text{Left} \mid s = L3) = 0.4$$

$$\pi(a = \text{Right} \mid s = L3) = 0.6$$

cf) 이해하기: 정책에서 확률적으로 행동하는 이유
확률적인 정책을 사용하여 항상 최적이라고 생각되는 행동만 하는 것이 아니라 때때로 다른 행동을 시도해 보면서 더 나은 정책을 학습할 수 있는 기회를 확보한다. 또한, 환경 역시 완전히 결정적인 것이 아니라 확률적으로 변할 가능성이 있어, 항상 같은 행동을 하는 결정적 정책은 예상치 못한 환경 변화에 유연하게 대응하기 어렵다.
즉, 확률적으로 행동하는 것은 단순히 랜덤하게 움직이기 위함이 아니라, 더 똑똑한 학습을 위해 필수적인 전략이다.

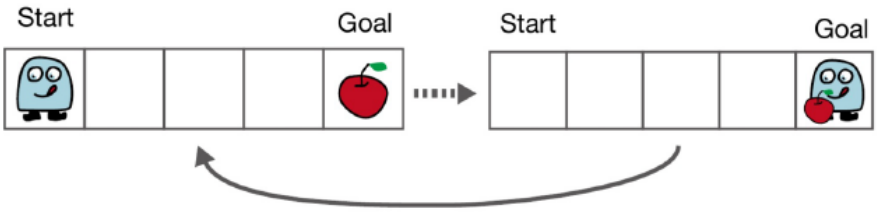
2.3. MDP의 목표

정리하면, 에이전트는 정책 $\pi(a|s)$에 따라 행동한다. 그 행동과 상태 전이 확률 $\rho(s'|s,a)$에 의해 다음 상태가 결정되고, 보상은 보상 함수 $r(s,a,s')$가 결정한다. 이 틀 안에서 최적 정책을 찾는 것이 MDP의 목표이다. 최적 정책이란 수익이 최대가 되는 정책이다.

MDP의 문제는 크게 '일회성 과제'와 '지속적 과제'로 나뉜다.

2.3.1. 일회성 과제와 지속적 과제

일회성 과제는 '끝'이 있는 문제이다. 예를 들어 바둑은 일회성 과제로 분류된다. 바둑은 승리/패배/무승부 중 하나로 귀결된다. 그리고 다음 대국은 돌이 하나도 놓여 있지 않은 초기 상태에서 새로 시작한다. 이러한 일회성 과제에서는 시작부터 끝까지의 일련의 시도를 에피소드라고 한다.



그림과 같이 어느 곳에 목표가 있고, 그 목표가 도달하면 끝이 난다. 후에는 다시 초기 상태에서 새로운 에피소드가 시작된다.

반면 지속적 과제는 '끝'이 없는 문제이다. 그 예로 재고 관리가 있다. 재고 관리에서의 에이전트는 얼마나 많은 상품을 구매할지를 결정한다. 판매량과 재고량을 보고 최적의 구매량을 결정해야 한다. 이런 유형의 문제는 끝을 정하지 않고 영원히 지속될 수 있다.

2.3.2. 수익

수익을 극대화하는 것이 에이전트의 목표이다.

시간 t에서의 상태를 S_t라고 한다. 그리고 에이전트가 정책 π에 따라 행동 A_t를 하고, 보상 R_t를 얻고, 새로운 상태 S_{t+1}로 전이하는 흐름이 이어진다. 이때 수익 G_t는 다음과 같이 정의된다.

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

수익은 에이전트가 얻는 보상의 합이다. 하지만 시간이 지날수록 보상은 γ(감마)에 의해 기하급수적으로 줄어든다. 이 γ를 할인율이라고 하며 0.0에서 1.0 사이의 실수로 설정한다. 예를 들어 할인율을 0.9로 설정하면 다음과 같다.

$$G_t = R_t + 0.9R_{t+1} + 0.81R_{t+2} + \cdots$$

할인율을 도입하는 주된 이유는 지속적 과제에서 수익이 무한대로 되지 않도록 방지하기 위해서이다.

또한 할인율은 가까운 미래의 보상을 더 중요하게 보이도록 한다. 예를 들어 오늘 1만원 받기와 10년 후에 2만원 받기 중 어느 쪽을 선택하겠습니까?의 문제에서 할인율 때문에 미래 보상이 기하급수적으로 줄어든다면 즉시 얻을 수 있는 보상에 더 큰 매력을 느낄 것이다.

cf) 이해하기: 할인율을 적용하는 이유
강화학습에서 수익은 단순히 모든 보상의 합을 더하는 것이 아니라, 각 보상에 할인율을 적용하여 가중치를 둔다.
할인율을 적용하는 이유는 미래 보상의 가치가 현재 보상보다 낮기 때문이다. 미래에 받을 보상은 현재보다 불확실하고, 실현될 가능성이 낮으며, 늦게 받을수록 가치가 떨어질 수 있다. (시간이 지날수록 보상의 현재 가치가 감소하기 때문에)
또한, 미래 보상을 그대로 더해버리면 먼 미래의 보상도 동일한 가중치를 가지게 되어 지나치게 장기적인 결과에 의존할 가능성이 있다.
또한, 할인율이 너무 크면 먼 미래의 보상을 중요하게 여겨 장기적인 이익을 추구하는 행동을 선호하게 되고, 할인율이 너무 작으면 가까운 미래의 보상을 중요하게 여겨 즉각적인 보상만 추구하는 행동을 하기 때문에 적절히 조절하면서 균형있게 고려하는 정책을 학습할 수 있어야 한다.

2.3.3. 상태 가치 함수

강화학습에서 에이전트는 '수익'을 극대화하는 것이 목표이다. 여기에서 주의할 점은 에이전트와 환경이 '확률적'으로 동작할 수 있다는 점이다. 에이전트는 다음 행동을 확률적으로 결정할 수 있고, 상태 역시 확률적으로 전이될 수 있다. 그렇다면 얻는 수익도 '확률적'으로 달라질 것이다. 비록 같은 상태에서 시작하더라도 수익이 에피소드마다 확률적으로 달라질 수 있다.

이러한 확률적 동작에 대응하기 위해서는 '수익의 기댓값'을 지표로 삼아야 한다. 상태 S_t가 s이고, 에이전트의 정책이 π일 때, 에이전트가 얻을 수 있는 기대 수익을 다음처럼 표현할 수 있다.

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s, \pi] \quad v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

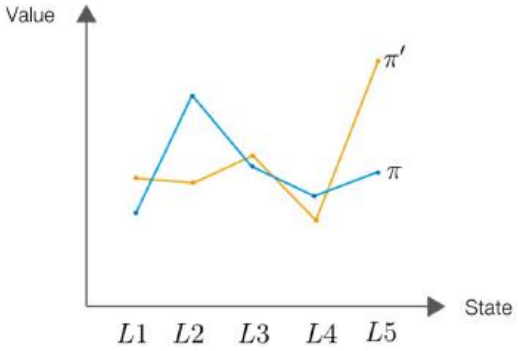
이처럼 수익의 기댓값을 $v_\pi(s)$로 표기하며 **상태 가치 함수**라고 한다.

식의 우변에서 에이전트의 정책 π는 조건으로 주어진다. 정책 π가 바뀌면 에이전트가 얻는 보상도 바뀌고 총합인 수익도 바뀌기 때문이다.

2.3.4. 최적 정책과 최적 가치 함수

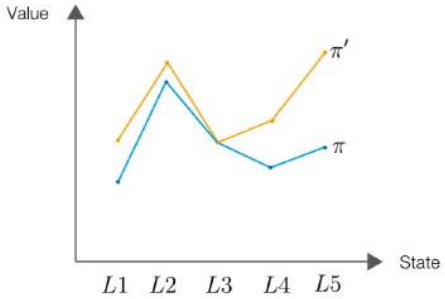
강화 학습의 목표는 최적 정책을 찾는 것이다.

먼저 두 가지 정책 π와 π'가 있다고 가정해보자. 이때 두 정책에서 상태 가치 함수 $v_{\pi}(s)$, $v_{\pi'}(s)$가 각각 결정된다. 모든 상태에서의 가치 함수가 그림과 같이 결정되었다고 가정한다.



그림에서 주목할 점은 상태에 따라 두 개의 상태 가치 함수가 좋을 때가 있고, 나쁠 때가 있다. 예를 들어 상태가 L1일 때는 정책 π'에 따라 행동하는 편이 수익의 기댓값이 더 높다. 하지만 L2일 때는 정책 π가 더 좋은 결과를 가져온다. 이처럼 상태에 따라 상태 가치 함수의 크고 작음이 달라지는 경우에는 두 정책의 우열을 가릴 수 없다.

두 정책의 우열을 가릴 수 있는 경우는 다음과 같다.



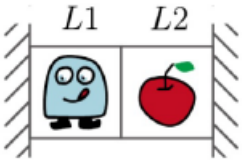
모든 상태에서 $v_{\pi'}(s) \geq v_{\pi}(s)$가 성립된다. 따라서 π'가 π보다 나은 정책이라고 할 수 있다. 어떤 상태에서 시작하더라도 얻을 수 있는 보상의 기댓값 총합이 π'쪽이 더 크거나 같기 때문이다.

이렇게 두 정책의 우열을 가리려면 하나의 정책이 다른 정책보다 '모든 상태'에서 더 좋거나 최소한 똑같아야 한다. 이 생각을 바탕으로 최적 정책은 다른 정책과 비교하여 모든 상태에서 상태 가치 함수의 값이 더 큰 정책이라는 뜻이다.

MDP에서는 최적 정책이 적어도 하나 존재한다는 사실이다. 그리고 그 최적 정책은 '결정적 정책'이다. 결정적 정책에서는 각 상태에서의 행동이 유일하게 결정된다. 최적 정책의 상태 가치 함수를 최저 상태 강치 함수라고 하고, 이를 v_{*}로 표기한다.

2.4. MDP 예제

MDP에 속하는 구체적인 문제를 살펴보자. 그림과 같이 두 칸짜리 그리드 월드가 있다.

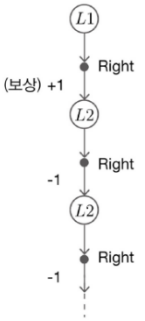


칸이 두 개이고, 좌우 끝은 벽으로 막혀 있다. 이 문제의 설정은 다음과 같다.

- 에이전트는 오른쪽이나 왼쪽으로 이동할 수 있다.
- 상태 전이는 결정적이다.
- 에이전트가 $L1$에서 $L2$로 이동하면 사과를 받아 +1의 보상을 얻는다.
- 에이전트가 $L2$에서 $L1$로 이동하면 사과가 다시 생성된다.
- 벽에 부딪히면 −1의 보상을 얻는다. 즉, 벌을 받는다. 예를 들어 에이전트가 $L1$에서 왼쪽으로 이동하면 −1의 보상을, $L2$에서 오른쪽으로 이동해도 마찬가지로 −1의 보상을 얻는다(이때 사과는 다시 생성되지 않는다).
- 지속적 과제, 즉 '끝이 없는' 문제다.

2.4.1. 백업 다이어그램

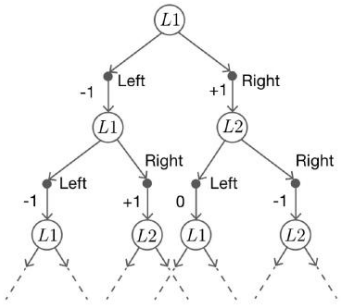
그림의 문제를 정리하기 위해 백업 다이어그램부터 그려본다. 백업 다이어그램은 '방향 있는 그래프'를 활용하여 '상태, 행동, 보상'의 전이를 표현한 그래프이다.



에이전트가 현재 상태에 상관없이 무조건 오른쪽으로 이동한다고 해보자. 이런 조건에서 에이전트의 행동과 생태 전이를 표현한 것이 왼쪽과 같다. 이 그림에서 시간은 위에서 아래로 흐른다.

그림에서 에이전트의 정책이 결정적이다. 즉 항상 정해진 행동을 취한다. 게다가 환경 상태 전이도 결정적이기 때문에 백업 다이어그램 전이는 일직선으로 뻗어 나간다.

하지만, 만약 에이전트가 50%의 확률로 오른쪽, 나머지 50%의 확률로 왼쪽으로 이동한다면 백업 다이어그램을 다음과 같이 그릴 수 있다.



그림과 같이 각 상태에서 오른쪽 이동과 왼쪽 이동 두 가지로 행동할 가능성이 있기 때문에 백업 다이어그램이 넓게 퍼져 나간다. 이번 문제에서는 더 단순한 그림(결정적 정책)에 집중하겠다. 즉, 환경의 상태 전이와 에이전트의 행동이 모두 결정적인 경우이다.

2.4.2. 최적 정책 찾기

그렇다면 두 칸짜리 그리드 월드에서 최적 정책은 무엇일까? 최적 정책은 결정적 정책으로 존재한다고 알려져 있다. 이번 문제에서는 상태와 행동의 가짓수가 적기 때문에 존재하는 모든 결정적 정책을 알아낼 수 있다. 상태와 행동이 각 2개씩이므로 결정적 정책은 총 $2^2 = 4$개가 존재한다.

	$s = L1$	$s = L2$
$\mu_1(s)$	Right	Right
$\mu_2(s)$	Right	Left
$\mu_3(s)$	Left	Right
$\mu_4(s)$	Left	Left

각 상태에서 취하는 행동을 정리한 표다. (결정적 정책이므로 각 상태에서 취하는 행동은 하나뿐이다.) 이 네 가지 정책 중 최적 정책이 존재하게 된다.

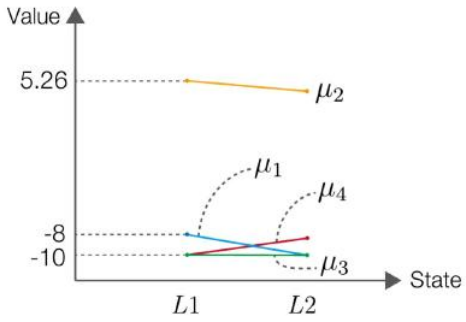
정책 μ_1의 상태 가치 함수를 계산해보자. 예를 들어 상태 L1에서 정책 μ_1에 따라 오른쪽으로 이동하면 보상 +1을 얻는다. 이후로는 오른쪽으로 이동하다가 벽에 부딪히고, 그때마다 -1의 보상을 얻는다. 이때 할인율을 0.9로 가정하면 상태 가치 함수는 다음처럼 계산할 수 있다.

$$\begin{aligned} v_{\mu_1}(S = L1) &= 1 + 0.9 \cdot (-1) + 0.9^2 \cdot (-1) + \cdots \\ &= 1 - 0.9(1 + 0.9 + 0.9^2 + \cdots) \\ &= 1 - \frac{0.9}{1 - 0.9} \\ &= -8 \end{aligned}$$

다음은 상태 L2에서의 가치 함수이다. 이번에는 오른쪽 벽에 계속 부딪히기 때문에 항상 -1의 보상을 얻게 된다. 따라서 상태 가치 함수는 다음처럼 계산할 수 있다.

$$\begin{aligned} v_{\mu_1}(s = L2) &= -1 + 0.9 \cdot (-1) + 0.9^2 \cdot (-1) + \cdots \\ &= -1 - 0.9(1 + 0.9 + 0.9^2 + \cdots) \\ &= -1 - \frac{0.9}{1 - 0.9} \\ &= -10 \end{aligned}$$

이렇게 정책 μ_1의 가치 함수를 구했다. 이처럼 다른 정책에도 모두 동일하게 진행하면 다음과 같은 결과를 얻을 수 있다.



그래프를 보면 정책 μ_2가 모든 상태에서 다른 정책들보다 상태 가치 함수의 값이 더 크다. 따라서 정책 μ_1가 최적 정책이라고 할 수 있다. 이 정책은 벽에 부딪히지 않고 오른쪽으로 갔다가 왼쪽으로 돌아오는 행동을 반복하여, 사과를 반복해서 얻어낸다.