Lecture 2

I. Introduction

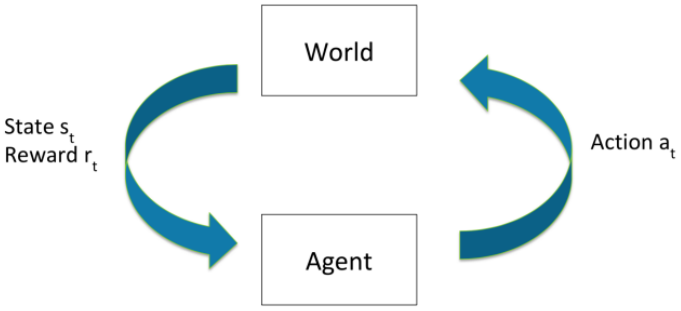

Image 01 : Basic Process of RL

지난 시간까지는 강화학습의 기본 요소인 Model, Value, Policy에 대해서 배워보았다. 요약하자면, World가 Agent에게 특정 state를 보내고, Policy에 따른 action을 토대로 Dynamics Model이 World를 바꾸면, Reward Model이 예상한 Reward(or Value)와 바뀐 state를 Agent에게 전달하는 구조를 배웠다. 이 때, 이 World는 Markov 성질을 만족하기 때문에 현재의 state만이 다음 state에 영향을 끼치며, 현재의 state는 observation을 통해 추론한다.

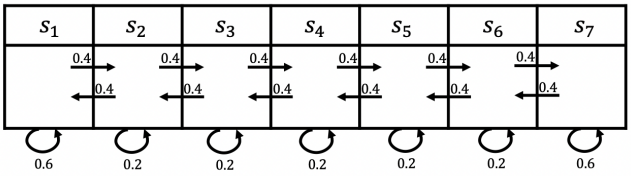

Image 02 : Mars Rover

이번 시간에는 <Image 02 : Mars Rover> 예시를 많이 사용한다. 위의 이미지는 개별 state마다 다음 state로 전이될 확률을 의미한다. 즉, s_4에서 s_3로 전이할 확률이 0.4, s_5로 전이할 확률이 0.5, s_4에 머무를 확률이 0.2임을 의미한다. 또한 Mars Rover는 s_1부터 s_7중 Random한 State에 착륙해 조사를 시작함을 유의해야 한다.

II. Markov Process

MP 개념

MP는 유한하거나 잠재적으로 무한한 State에서, 이전 state에 따른 다음 state를 추정하는 Dynamics Model이며, reward와 action에 대한 개념이 없다.

$$P = \begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$

MP에서는 다음의 단어의 정의가 필요하다.
S는 유한 공간 속의 state의 집합이다.
P는 모든 state에 대해서 $p(s_{t+1} = s'|s_t = s)$를 특정하는 전이 확률이며 위의 행렬처럼 적는다. 즉, s에서 s'으로 전이될 확률을 사전에 정의한 값으로 변하지 않는다.

만약 특정 state에서 episode를 시작한다면, 전이 확률을 통해서 다양한 trajectory가 나올 수 있으며, 실제 World에서는 그 중 하나의 trajectory를 수행한다. MP에서의 trajectory는 단순한 state의 연속을 의미한다.

MP 예시

$$P = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.2 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}$$

<Image 02 : Mars Rover>의 전이 행렬은 위와 같이 적을 수 있다. 만약 현재 state가 s_2라면,
$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot P$를 통해 다음 state가 $\begin{bmatrix} 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \end{bmatrix}$임을 알 수 있다. 즉, s_2에서 시작한다면 다음 state에서 s_1으로 갈 확률이 0.4, s_2에 머무를 확률이 0.2, s_3로 갈 확률이 0.4라는 뜻이다.

trajectory는 $s_4, s_5, s_6, s_7, s_7, ...$, $s_4, s_3, s_3, s_2, s_2, s_1, ...$ 등의 다양하게 발생할 수 있는 state의 Seqeunce를 의미한다.

III. Markov Reward Process

개념

MRP는 MP에 Reward를 추가한 개념으로, 다음의 단어에 대한 정의가 필요하다.
R은 MRP에서는 특정 state마다 얻을 수 있는 reward이다.
γ는 미래에 얻을 수 있는 reward에 대한 할인 계수이며, 0에 가까울수록 즉각적 reward에 집중하겠다는 의미이다.
Horizion은 episode마다 몇 번의 time-step을 허용할지에 대한 지표로, agent가 얼마나 오랫동안 존재할지에 대한 지표이다 .
G_t는 Return으로, 현재 state부터 horizon step까지 가며 γ를 곱해 얻은 reward의 총합이다.
V는 Value로, 예상되는 Return의 평균이며, 결정론적 관점에서는 Return과 Value가 동일하다.

MRP에서 Value 계산하기

앞으로는 “N번 time-step이 변할 때, 얻을 수 있는 Value는 얼마인가?”에 대한 내용이다. reward 함수 R, 전이 확률 P, 할인 계수 γ가 고정되어있을 때, Value 계산법은 총 3가지가 있다.

Method 1 : Monte Carlo Simulation

state마다 여러개의 episode를 만들어, episode마다의 G_t를 추정하여 평균을 내는 방법이다.
이 방법의 경험적 평균과 실제 평균이 유사해 질 수 있으며 빠르게 수렴할 수 있으나, Markov 특성을 사용하지 않았고 충분히 많은 Episode를 만들었는지는 의문이라는 것이다.

Method 2 :Analytic Solution

$$V(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s)V(s') \cdots (1)$$
$$V = R + \gamma PV \cdots (2)$$
$$(I - \gamma P)V = R \cdots (3)$$
$$V = (I - \gamma P)^{-1}R \cdots (4)$$

우선, horizon이 유한이고 $\gamma < 1$을 만족할 때 수식 (1)을 정의할 수 있다. 즉, $V(s)$는 즉각적 Reward와 미래에 얻을 수 있는 value에 할인계수를 곱한 값의 합이다. 그리고 이 V에서 Reward, γ, P가 모두 정의된 값이기에 수식 (2)와 같이 정리할 수 있으며, V에 대해서 수식 (4)로 정리할 수 있다. 이 방법은 Monte-Carlo보다 V의 값을 더 잘 추정할 수 있지만, $O(|S|^3)$만큼 계산해야하며 Infinite Horizon에서는 불가능하다는 것이다.

Analytic Solution에서 V(s')에서의 R(s)이 언제까지 반영되는가?

Method 3 : Iterative Algorithm

$$V_k(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s)V_{k-1}(s'), \quad \forall k = 1, ..., H$$

horizon이 무한이고 $\gamma < 1$을 만족할 때 성립한다. Iterative Solution은 Analytic Solution과 매우 유사하게 생겼지만, 유한 공간과 무한 공간의 차이가 있다. Iterative Algorithm은 Markov 성질을 이용하여 $k = 1$부터 증가하며, $|V_k - V_{k+1}| < \epsilon$을 만족할 때까지 반복한다. 따라서 무한한 공간에서 Value가 ϵ보다 작아지는 안정적인 공간까지 확장할 수 있다. 이 방법은 계산 비용이 $O(|S|^2)$으로 개별적인 update가 저렴하고, MDP에서 Action을 고려했을 때 더 유용하다. 하지만 ϵ를 크게 설정하면 V가 적합하지 않을 수 있으며, ϵ을 작게 설정하면 계산 비용이 증가할 수 있다.

예시

<Image 02 : Mars Rover>를 다시 살펴보겠다. s_1일 때는 +1의 reward를, s_7일 때는 +10의 reward를 받으며, $\gamma = 0.5$, horizon이 4인 경우를 생각해보겠다. 같은 초기 state이더라도 굉장히 다양한 epsiode가 나올 수 있다. 이 때 다양한 episode에 대한 Reward를 계산하면 다음과 같다.

$s_4, s_5, s_6, s_7 : 0 + (0.5) * 0 + (0.5)^2 * 0 + (0.5)^3 * 10 = 1.25$
$s_4, s_4, s_5, s_4 : 0 + (0.5) * 0 + (0.5)^2 * 0 + (0.5)^3 * 0 = 0$
$s_4, s_3, s_2, s_1 : 0 + (0.5) * 0 + (0.5)^2 * 0 + (0.5)^3 * 1 = 0.125$

V는 이렇게 만든 모든 episode의 reward를 평균낸 값과 동일하다. 위의 계산법 중 Analytic Solution을 사용하면 $V = \begin{bmatrix} 1.53 & 0.37 & 0.13 & 0.22 & 0.85 & 3.59 & 15.31 \end{bmatrix}$를 얻을 수 있다. 즉, s_1의 예상된 reward의 평균은 1.53이며, s_7의 예상된 reward의 평균은 15.31이다.

IV. Markov Decision Process

개념

MDP는 MRP에 Action을 추가한 개념으로, 다음의 단어에 대한 정리가 필요하다.
A : action a에 대한 집합을 의미하며, $a \in A$를 만족한다.
$Policy$: π : 확률론적인 관점에서 action에 대한 분포를 의미하거나, 결정론적으로 state에 따른 action을 정해주는 mapping 함수를 의미한다.
P : 각각의 action에 따른 전이 확률 모델로 $P(s_{t+1} = s' | s_t = s, a_t = a)$로 적을 수 있다.
$P^{\pi}(s'|s)$: $\sum_{a \in A} \pi(a|s) P(s'|s,a)$로, 어떤 state에서 특정 policy π를 따라 결정한 action을 가중치로 가지는 전이 행렬을 의미한다.
R : Reward 함수로 $R(s_t = s, a_t = a)$로 적을 수 있으며, 이 값은 $\mathbb{E}(r_t | s_t = s, a_t = a)$를 의미한다.
R^{π} : $\sum_{a \in A} \pi(a|s) R(s,a)$로, 어떤 state에서 특정 policy π를 따라 결정한 action의 reward를 의미한다.
k : 특정 time-step을 의미합니다.
$V_k^{\pi}(s)$: State Value Function : 특정 time-step k에서 state가 s일 때, policy π를 따르면 얻을 수 있는 discounted reward의 sum 합을 의미한다.
$Q_k^{\pi}(s,a)$: State-action Value Function : V_k^{π}와 유사하게 생겼으며, 같은 역할을 한다. 대신 Q 함수는 state와 action을 함께 고려한다. 즉, 특정 time-step k에서 state가 s이고 action이 a일 때, policy π를 따르면 얻을 수 있는 discounted reward의 sum 합을 의미한다.

이 모든 개념을 합쳐서 MDP는 (S, A, P, R, γ)로 표기한다.

Policy Evaluation

MDP에 π가 얼마나 좋은 policy인지를 평가하고자 한다. 이 때, MRP에서 사용한 V 추정 방법론 3가지를 사용할 수 있다. 그 중 Iterative Solution을 응용해보자. 초기 V를 0으로 설정하고, 모든 state에 대해 에 k를 증가시키며 $V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^{\pi}(s')$를 반복해 V^{π}를 업데이트하면 된다. 이 식을 통해 k가 증가함에 따라, s에 대한 V가 바뀌고 있으며, time-step이 진행됨에 따라 미래의 reward의 정보가 앞으로 넘어오고 있다. 정보의 흐름의 관점에서 생각해보면서 아래의 Mars Rover 문제를 해결하겠다.

$k = 1$이고 $\gamma = 0$일 때의 V는 어떻게 구성될까? 이 질문은 $V_{k=1}^{\pi}(s)$를 물어보는 것으로,
$\begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & +10 \end{bmatrix}$이다. 왜냐하면 γ가 0이기 때문에 즉각적인 reward에만 집중하겠다는 의미이기 때문이고, 미래의 정보가 현재 state로 역전파되지 않는다.

policy는 $\pi(s) = a_1, \forall s$를, $p(s_6 | s_6, a_1) = 0.5, p(s_7 | s_6, a_1) = 0.5$를 만족한다. $k = 2$이고 $\gamma = 0.5$일 때, s_6에서의 V는 얼마일까? 이 질문은 $V_{k=2}^{\pi}(s_6)$를 물어보는 것으로, $0 + \gamma[0.5 \cdot V(s_6) + 0.5 V(s_7)] = 0 + 0.5[0 + 0.5 \cdot 10] = 2.5$임을 알 수 있다. 이 때는 time-step이 진행됨에 따라 2-step이 었던 reward의 흐름이 점차 더 멀리 퍼져나갈 수 있다.

이 내용은 V를 평가하는 것이 아닌 π를 평가하는 것이다. 즉, 위의 내용은 특정 π를 따라갔을 때의 V로, 개별 π에 대한 성능임을 명심해라.

MDP Control

Policy를 이용해 MDP를 통제해보자. 사실 궁극적으로는 agent가 policy를 학습했으면 좋겠지만, 우선은 여러 policy 중 최적의 policy를 계산하는 방법을 알아보자. 최적의 policy는 $\pi^*(s)$로, $\underset{\pi}{argmax} V^\pi(s)$ 를 의미한다.

최적의 V는 하나만이 존재하지만, 최적의 V를 만드는 Policy $\pi^*(s)$는 여러개 일 수 있다. 또한 무한한 horizon에서의 MDP를 따르는 최적 policy는 결정론적이다. 필자는 가장 좋은 policy는 자신의 action에 대한 확신을 가지고 결정론적으로 선택하기 때문이라고 생각했다.

결정론적 관점에서의 policy는 개별 state마다 action을 특정한다. 따라서 S와 A를 따르는 policy는 $|A|^{|S|}$개 이다. 예시로 Mars Rover의 경우 3개의 action이 존재한다면, 총 3^7개 만큼의 policy가 존재한다는 것이다. 현실 세계에서는 모든 state에서 모든 action을 할 수는 없으며, 특정 state에 맞는 action이 존재한다. 즉, 공부 중인 state에서 날아간다는 action이 나올 수 없는 것과 비슷하다. 하지만 우선은 그런 경우를 배제해보도록 한다.

V. Find Optimal Policy π^*

유한 S와 유한 A, 안정된 R과 안정된 P가 있다면, 최적의 V는 단일 값으로 보장되고, 그 값을 가지게 하는 policy를 π^*라고 한다. 그렇다면 $|A|^{|S|}$개의 모든 Policy 중 최적의 Policy를 어떻게 찾으면 좋을지 고민해보자.수업에서는 Policy Search와 Policy Iteration, Value Iteration 3가지를 다루고 있다.

Policy Search

Policy Search는 모든 가능한 경우의 수 $|A|^{|S|}$에 대해서, policy마다 V를 계산하고, 그 중에서 가장 좋은 V^π와 π^*를 찾는 방법이다. 모든 경우의 수를 탐색하기에 좋다는 장점이 있지만, 계산 비용이 $O(|A|^{|S|})$로 매우 클 뿐만 아니라 S가 매우 크다면 불가능한 방법이다. 사실 필자는 이 방법은 policy를 검색하는 방법이지, 탐색하는 방법이 아니라고 생각하기에, 대안이 없거나 전반적인 flow를 파악하기 위해 사용한다고 본다.

Policy Iteration(PI)

Policy Iteration(이하 PI)은 최적의 π가 결정론적 policy를 따르며,V_0는 0으로 두고, Iteration을 거치면서 policy가 점점 나아진다고 본다. Iteration은 0 부터 증가하며, $V^{\pi_i} \leq V^{\pi_{i+1}}$을 만족하며, $||\pi_i - \pi_{i+1}||_1$가 0이 되었을 때 멈춘다. 멈추었을 때의 π_i를 optimal policy π^*으로, 이 때의 값을 V^*으로 본다.

개별 iteration에서는 모든 state에 대해 Policy Evaluation을 통해 $V(s)$를 구해주고, 변경된 $V(s)$를 바탕으로 Policy Improvement을 통해 그 state의 최적 action을 정해준다.

Value Iteration(VI)

Vaue Iteration(이하 VI)에서도, 최적의 π가 결정론적 policy를 따르며 V_0는 0으로 두고, Iteration을 거치며 policy가 점점 나아진다고 본다. Iteration을 0부터 증가시키며, 모든 state에 대한 $V(s)$가 수렴할 때까지 반복한다. 개별 Iteration에서는 주어진 s에서 가장 V가 높아지도록 a를 선택하고, 그렇게 선택된 action을 π에 반영한다.

의문점들

항상 $V^{\pi_i} \leq V^{\pi_{i+1}}$을 성립하는가? 이게 옳지 않다면 위의 PI는 사실 불가능하다. 하지만 다행히도 항상 성립한다. $V^{\pi_i} \leq V^{\pi_{i+1}}$이 성립한다는 것은, 특정 s에서도 $V^{\pi_i}(s) \leq V^{\pi_{i+1}}(s)$를 성립한다는 것이다. s에서 π_i보다 π_{i+1}을 따랐을 때, 정의에 의해 V가 더 크거나 같을 것이고, s'일 때도 π_i보단 π_{i+1}을 따랐을 때 V가 더 크거나 같을 것이다. 이를 S에 대해서 반복한다

면, 결과적으로 V^{π}는 $V^{\pi_{i+1}}$ 보다 작거나 같을 수 밖에 없다. 따라서 PI에서 iteration을 거치면 거칠수록 점진적으로 더 발전한 policy를 고를 수 있다. 이를 **monotonic improvement**라고 부르며, 증명은 Appendix를 참고하길 바란다.

그렇다면 $V^{\pi_i} \leq V^{\pi_{i+1}}$이 발산하지 않는가? 즉 $0 \leq 1 \leq 2 \leq ...$ 이면, 결국에는 끝 없이 Value가 성장하기에 계속해서 Iteration을 반복해야 할 것이다. 다행히 이 내용은 **Bellman Backup Operator**가 Contraction을 만족하기 때문에, $||V^{\pi_i} - V^{\pi_{i+1}}||$는 항상 수렴한다. 또한 Bellman Bakcup Operator는 Banach Fixed Point Theorem을 만족하기 때문에 최적의 V는 하나만을 가진다. 자세한 내용은 Appendix를 참고하길 바란다.

마지막으로 언제까지 iteration을 반복해야하는가? 우선 iteration은 policy가 바뀌지 않을 때 까지 반복해야 하며, 모든 policy에 대해 탐색하는 $|A|^{|S|}$번 만큼 증가할 수 있다.

Summary

이번 시간에는 특정 state에서 action을 했을 때, 그 action이 불러올 다음 state의 분포와 Value를 높이기 위한 결정 방법을 알아보았다. 각각의 키워드는 “Markov Process”, “Markov Reward Process”, “Markov Decision Process”, “Find Optimal Policy π^*”이며 각각에 대해서 간단히 요약하면 다음과 같다.

MP는 “Markov 성질을 만족하는 이산 확률 과정"으로, 현재 state에 따라 다음의 state를 확률적으로 결정한다는 개념이다. 결정론적 관점에서는 미래가 정해진 여러 결과 중 하나만이 되며, 확률론적 관점에서는 다음 state에 대한 가능성이 반환된다. 또한 MP에서는 고정된 확률과 초기 state에 따라 완전히 결정되며, agent가 개입할 여지가 없다.

MRP는 MP에서 Reward와 Discount Factor를 추가한 개념으로, 각각의 state에 대한 Reward(or Value)를 계산할 수 있으며, Reward를 계산하는 방법에는 Monte-Carlo simulation, Analytic Solution, Iterative Alrofithm이 있었다. MRP에서는 여전히 agent가 개입할 수 있는 여지가 없다.

MDP는 MRP에서 Action을 추가한 개념으로, Value를 최대화 할 수 있도록 agent가 Policy를 통해 Action을 선택할 수 있게 됐다. MDP부터는 agent의 개입 여지가 생겼기에, agent의 action에 따른 전이확률이 달라진다. MDP의 최종 목표도 State-Action Value Q를 최대화하는 것이기에, MRP에서 사용한 방법론 3가지를 변형해 최적의 Policy를 찾는데 사용할 수 있다.

최적의 Policy를 찾기위한 Policy Iteration은 현재 주어진 상태에서 최적의 Policy가 무엇인지를 점진적으로 갱신하는 방법이며, Value Iteration은 현재 내가 어떤 action을 취해야 Value가 가장 높아질지를 선택한 후 그 action을 policy에 반영하는 방법이다.

필자는 다음과 예시를 들며 마치도록 하겠다.
MP는 과거 고려시대의 신분 사회로, 자신(agent)의 신분(현재 state)에 따라 내 미래의 신분(미래 state)이 결정되는 세상이며, 각 신분마다 신분이 변동할 확률이 정해져있다(유지, 하락, 상승).
MRP는 reward를 신분에 따른 소득으로 고려했을 때, 각 신분에 따른 소득은 높을수록 많이 받는다. 하지만 5살때까지는 자신의 신분에 만족하며 살아왔다(no action).
MDP는 소득을 올리기 위한 나의 action이 추가되었다. 즉, 6살때부터 15살까지(finite horizon) 소득을 올리기 위해 하루에 3가지만을 선택하여 놀기, 잠자기, 공부하기, 운동하기의 4가지(action)을 하려 한다. 그리고 내 목표는 최단기간에 가장 소득을 높이는 방법을 선택하는 것이다.
최적의 policy를 찾기 위해 매일매일 4가지의 action을 조합해서 사용해보며, 미래의 reward에 0.5의 γ를 가지고 미래를 그려보았다. 그랬더니 잠을 줄이며 공부하기와 잠과 운동을 적절히 배분한 것이 공동으로 최고의 소득을 가질 수 있도록 도와주었다.