교재 01, 02장 복습

2024/09/04

01. 강화학습 개요

1 - 강화학습의 개념

- **행동심리학자 스키너의 "강화(Reinforcement)"**

  배고픈 쥐가 상자 안을 돌아다니다가 우연히 지렛대를 누르면 먹이가 나온다. 그러한 상황이 계속되다보면 왜 그런지 이해하지는 못하더라도 지렛대를 누르면 먹이가 나온다는 사실을 알게 된다. 그리고 그 좋은 결과를 얻기 위한 행동을 자주 하게 된다.

  이러한 시행착오 학습을 통해 동물이 좋은 보상을 얻을 수 있는 행동을 점점 더 많이 하는 것을 '강화'라고 한다.

- **머신러닝과 강화학습**

  머신러닝은 지도학습, 비지도학습, 강화학습으로 나뉜다.

  지도학습은 정답을 알고 있는 데이터를 이용해 컴퓨터를 학습시킨다. 정답과 예측값의 차이를 최소화하는 방향으로 학습한다. 반면 비지도학습은 정답이 없이 주어진 데이터로만 학습한다. 비슷한 데이터끼리 묶어주는 식으로 학습한다.

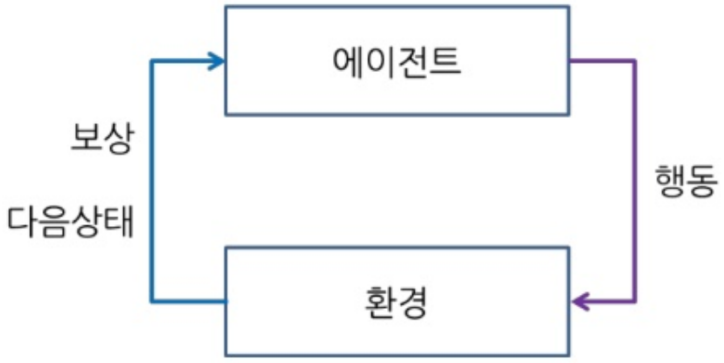
  이것들과 다르게 강화학습은 주어진 데이터에 대해 학습하는 방식이 아니다. **강화학습은 보상(Reward)을 통해 학습한다.** 보상은 컴퓨터가 선택한 행동(Action)에 대한 환경(Environment)의 반응으로, 간접적인 정답의 역할을 한다. 강화학습을 수행하는 컴퓨터는 보상을 얻게 하는 행동을 점점 더 많이 하도록 학습한다.

- **강화학습**

  강화학습을 통해 스스로 학습하는 컴퓨터를 **에이전트**라고 한다.

학습 초기의 에이전트는 자신을 둘러싼 모든 것, **환경**에 대한 아무런 정보가 없다. 환경에서의 자신의 **상태**를 인식한 후 거의 무작위로 **행동**을 수행한다. 그리고 환경으로부터 그에 따른 **보상**과 **다음 상태**에 대한 정보를 받는다. 보상을 바탕으로 특정 상황에서 특정 행동을 하는 것에 값을 매기고 갱신하며 **가장 높은 보상을 받을 수 있는 행동 집합인 최적 정책을 구하고자 한다.**

이 때 보상은 양수로 설정할 수도 있고 음수로 설정할 수도 있다. 상(+)과 벌(-)을 적절하게 융합하여 학습이 효과적으로 이루어지도록 하는 것이 중요하다.



2 - 강화학습의 구성 요소

1. **상태 state**

   에이전트의 정적인 요소뿐만 아니라 에이전트가 움직이는 속도와 같은 동적인 요소 또한 상태로 표현될 수 있다. 에이전트가 상태를 통해 상황을 판단해서 행동을 결정하므로 상태는 충분한 정보를 제공해야 한다. 상태의 정의,표현이 중요하다.

2. **행동 action**

   에이전트가 어떠한 상태에서 취할 수 있는 행동이다. 일반적으로 모든 상태에서 같은 행동 집합이 존재한다. 예를 들어, 상, 하, 좌, 우로 움직일 수 있는 에이전트가 있다면 이 에이전트는 모든 상태에서 상, 하, 좌, 우로 움직일 수 있다. 하지만 에이전트는 학습하면서 어떤 행동이 좋은 행동인지 점차 알게 되므로 각 행동들을 할 확률은 달라질 것이다.

3. **보상 reward**

   에이전트의 행동에 대해 환경이 전달해주는 값이다. 사실상 에이전트가 학습할 수 있는 유일한 정보이다. 특정 상태에서 특정 행동이 좋은지 나쁜지를 알 수 있고, 간접적으로

규칙을 학습하는 지표가 된다. 보상은 에이전트가 아니라 환경의 일부이므로 에이전트는 어떤 상황에서 얼마의 보상이 나오는지 미리 알 수 없다.

4. **정책 policy**

   모든 상태에 대해 에이전트가 어떤 행동을 해야 하는지 정해 놓은 것이다. 정책은 가능한 행동들이 각각 선택될 확률을 지정하고, 그 확률들은 행동 선택의 지표로 사용된다.

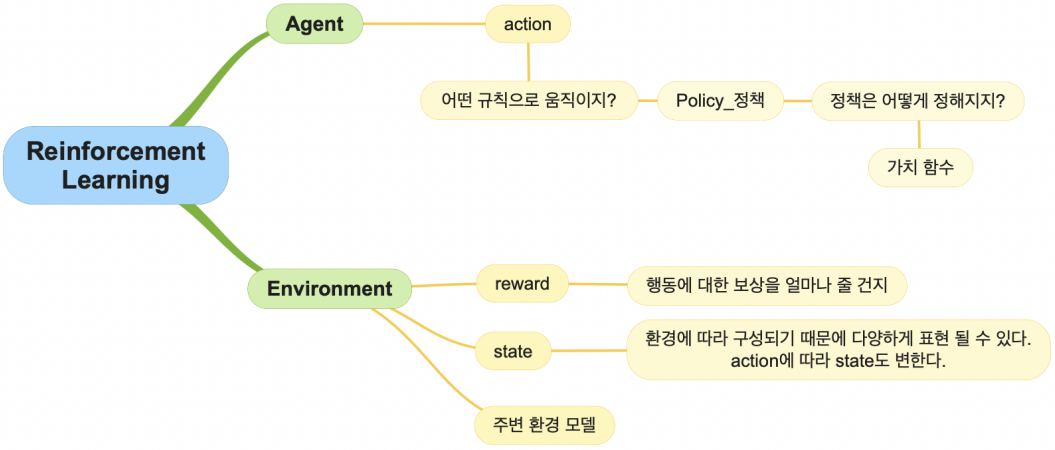
   제일 좋은 정책은 **최적 정책**이라고 하며, 에이전트는 최적 정책에 따라 행동했을 때 보상의 합을 최대로 받을 수 있다.

5. **가치 함수 value**

   가치란 특정 상태의 시작부터 일정 시간 동안 에이전트가 기대할 수 있는 보상의 총량이다. 따라서 가치는 장기적인 관점에서 상태를 표현할 수 있고, 이는 행동 보상의 합을 최대로 만들고자 하는 강화학습의 학습목표와 밀접한 관련이 있다. 따라서 일반적으로 에이전트는 보상 대신 가치를 기준으로 행동을 결정한다고 볼 수 있다.

6. **주변 환경에 대한 모델 model**

   모델은 환경의 변화를 모사하여 환경이 어떻게 변할지를 추정하게 해준다. 현재 상태와 행동으로부터 다음 상태와 보상을 예측한다. 이런 모델을 사용하는 경우를 '모델 기반(model-based) 방법'이라고 한다. 모델 없이 시행착오를 통해서만 학습하는 경우는 'model-free 방법'이라고 한다.



02. 강화학습 기초

1 - 순차적 행동 결정 문제

강화학습은 순차적으로 행동을 결정해야 하는 문제를 다룬다.

강화학습에서 에이전트는 한 타임스텝마다 자신의 행동을 결정한다. 이 결정은 다음 선택에 영향을 미치므로 현재 상태는 바로 이전 상태와 행동에 의존적이다. 이처럼 순차적으로 선택을 하고, 그 선택을 바탕으로 또 다른 선택을 하는 과정을 순차적 행동 결정 문제라고 한다.

순차적 행동 결정 문제를 풀기 위해서는 컴퓨터인 에이전트가 이해할 수 있도록 문제를 수학적으로 정의해야 한다. 이 때 사용하는 방법이 **MDP (Markov Decision Process)**이다.

2 - MDP의 구성 요소

1. **상태**

   S : 에이전트가 관찰 가능한 상태의 유한집합

   상태 == "자신의 상황에 대한 관찰"

   S_t : 시간 t일 때의 상태

   MDP에서 상태는 시간에 따라 확률적으로 변하므로 시간 t에 에이전트가 있을 상태는 전체 상태 집합 S의 요소 중 하나가 될 것이다. 따라서 확률변수로써 $S_t \in S$ 로 표현될 수 있다.

   $S_t = s$: 시간 t에서의 어떤 상태 s

2. **행동**

   A : 에이전트가 상태 S_t에서 할 수 있는 가능한 행동의 집합

   보통 에이전트가 할 수 있는 행동은 모든 상태에서 같다. (행동 집합이 동일하다.)

   $A_t = a$: 시간 t에서의 특정 행동 a

   특정 시간에 에이전트가 어떤 행동을 할지 정해져 있지 않으므로 행동도 확률변수이다.

3. **보상 함수**

: 시간 t일 때 상태가 $S_t = s$이고 그 상태에서 행동 $A_t = a$를 했을 경우에 받을 보상에 대한 기댓값

: $r(s,a) = E[R_{t+1}|S_t = s, A_t = a]$

- 기댓값 : 어떤 정확한 값이 아니라 나오게 될 숫자에 대한 예상
- 보상함수를 기댓값으로 표현하는 이유?

  ➡️ 보상을 에이전트에게 주는 것은 환경이고, 환경에 따라서 같은 상태에서 같은 행동을 취하더라도 다른 보상을 줄 수 있기 때문이다.
- R_{t+1}인 이유?

  ➡️ 보상을 에이전트가 알고 있는 것이 아니라 환경이 알려주는 것이기 때문이다. 에이전트가 시간 t의 상태 s에서 행동 a를 하면 환경은 보상을 다음 타임스텝인 $t+1$에 준다.

4. **상태 변환 확률**

   : 에이전트가 상태 s에서 행동 a를 해서 s'으로 가게 되는 확률

   (s' : 다음 타임스텝에 에이전트가 갈 수 있는 어떤 특정한 상태)

   바람이 불거나 넘어지는 등 반드시 바로 앞에 있는 상태에 도달하지 못할 수도 있다. 이처럼 상태의 변화에는 확률적인 요인이 들어가므로, 이를 수치적으로 표현한 것이 상태 변환 확률이다.

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

   == **환경의 모델**

   환경은 에이전트가 행동을 취하면 상태 변환 확률을 통해 에이전트가 갈 다음 상태를 알려준다.

5. **할인율**

   에이전트가 항상 현재에 판단을 내리므로, 현재에 가까운 보상일수록 더 큰 가치를 가진다.

   ⇒ 같은 보상이면 나중에 받을수록 가치가 줄어든다.

   할인 == 미래의 가치를 현재의 가치로 환산하는 것

   할인율 == 시간에 따라 할인하는 비율

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

현재의 시간 t로부터 시간 k가 지난 후 받는 보상의 할인율을 고려한 현재 가치

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

(보상은 행동의 1 타임스텝 이후에 받으므로 γ^{k-1}만큼 할인되는 것이다.)

정책

: 모든 상태에서 에이전트가 할 행동

정책을 상태가 입력으로 들어오면 행동을 출력으로 내보내는 일종의 함수라고 생각해도 좋다.

에이전트가 상태 $S_t = s$에 있을 때, 가능한 행동 중에서 $A_t = a$를 할 확률이라는 수식으로 나타낼 수 있다.

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

> 💡 정책에서 특정 상태에 어떤 선택을 하는지는 일반적으로 **탐욕적**으로 정해진다. 탐욕적인(**greedy**) 선택은 '선택지 중 가장 큰 값을 선택하는 방식'이다. 따라서 정책에서 탐욕적인 선택은 '모든 상태마다 가장 가치가 높은 행동을 선택하는 것'을 의미한다.

에이전트가 MDP를 통해 최적 정책을 찾는 방법

- 특정 상태에 에이전트가 있다고 가정할 때, 에이전트는 좋은 행동을 하기 위해 현재 상태에서 앞으로 받을 보상들을 고려해야 한다.
- 앞으로 받을 보상에 대한 개념 : 가치함수

3 - 가치함수

시간에 따른 보상을 단순히 전부 더한다면 다음과 같은 문제가 생긴다.

1. 현재 받은 보상이나 미래의 보상이나 똑같이 취급하므로, 현재 100의 보상을 받은 경우와 시간이 흐른 미래에 100의 보상을 받은 경우를 구분할 수 없다.
2. 100이라는 보상을 1번 받는 것과 20이라는 보상을 5번 받는 것을 구분할 수 없다.

3. 시간이 무한대일 경우 매 타임스텝마다 0.1의 보상을 받는 경우와 1의 보상을 받는 경우가 둘 다 보상의 합은 무한대로 동일하여 수치적으로 두 경우를 구분할 수 없다.

따라서 좀 더 정확하게 상태의 가치를 판단하기 위해 할인율이라는 개념을 사용하는 것이다.

- **반환값**

  : 할인율을 적용한 보상들의 합

  에이전트가 실제로 종결 상태에 다다르는 시점까지(= 한 에피소드가 끝날 때까지) 환경을 탐험하며 받은 보상의 합

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

  $= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

> 💡 에피소드 : 유한한 에이전트와 환경의 상호작용

에피소드가 끝난 후, 에이전트가 보상을 정산하는 것이 반환값이다.

MDP로 정의되는 세계에서 에이전트와 환경의 상호작용은 불확실성을 내포하고 있다. 그래서 특정 상태의 반환값은 에피소드마다 다를 수 있다.

따라서 **반환값에 대한 기댓값**으로 특정 상태의 가치를 판단한다. → **가치함수**

(현재 에이전트가 주어진 상태에 있는 것이 얼마나 좋은지를 추정하는 함수)

<입력>				<출력>
상태	→	가치함수	→	받을 보상의 합

- **가치함수 (상태 가치함수)**

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

  $= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s]$

  $= E[R_{t+1} + \gamma G_{t+1}|S_t = s]$

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

  - 확률변수인 보상들의 합으로 나타낸 반환값도 확률변수이다. 하지만 가치함수는 특정 양을 나타내는 값이므로 소문자로 표현한다.

- 상태의 가치를 고려하는 이유?

  ➡️ 현재 에이전트가 갈 수 있는 상태들의 가치를 안다면 그중에서 가장 가치가 제일 높은 상태를 선택할 수 있기 때문이다.

> 💡 에이전트가 앞으로 받을 보상에 대해 생각할 때 **정책**을 고려해야 하는 이유 : 상태에서 상태로 넘어갈 때 에이전트는 무조건 **행동**을 해야 하고, 각 상태에서 행동을 하는 것이 에이전트의 정책이기 때문이다.

> 💡 에이전트가 현재 상태에서 다음에 어떤 상태로 갈지 결정하는 것 : 에이전트가 정책에 따라 선택할 행동 + 상태 변환 확률

정책을 고려한 가치함수

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

$= \sum_{a \in A} \pi(a|s) \sum_{s', r} P(s', r|a, s)[r + \gamma v_\pi(s')]$

== **벨만 기대 방정식** (현재 상태의 가치함수와 다음 상태의 가치함수 사이의 관계를 말해주는 방정식)

- **큐함수 (행동 가치함수)**

  : 어떤 상태에서 어떤 행동이 얼마나 좋은지 알려주는 함수

  큐함수도 벨만 기대 방정식의 형태로 나타낼 수 있다.

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

  - 가치함수와 큐함수 사이의 관계식
    - 각 행동을 했을 때 앞으로 받을 보상인 큐함수 $q_\pi(s, a)$를 정책 $\pi(a|s)$에 곱한다.
    - 모든 행동에 대한 큐함수와 정책을 곱한 값을 더하면 가치함수가 된다.

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

💡 에이전트는 큐함수를 기준으로 어떤 행동이 가장 좋은지를 안다.

- **벨만 기대 방정식**

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

  - 한 번에 모든 것을 계산하는 것이 아니라, 값을 변수에 저장하고 루프를 도는 계산을 통해 참 값을 알아나가는 것이다.
  - 식의 왼쪽 값을 오른쪽 수식의 값으로 대체 (현재 가치함수 값을 업데이트) ⇒ 기댓값(E) 계산해야 한다!
  - 기댓값에는 어떠한 행동을 할 확률(정책)과 그 행동을 했을 때 어떤 상태로 가게 되는 확률(상태 변환 확률)이 포함되어 있다.

    ➡️ 계산 가능한 벨만 방정식

    : $v_\pi(s) = \Sigma_{a\in A}\pi(a|s)\big(r_{(s,a)} + \gamma\Sigma_{s'\in S}P^a_{ss'}v_\pi(s')\big)$

    - 환경의 일부인 상태 변환 확률($P^a_{ss'}$)은 환경을 만드는 사람이 설정할 수 있다.
    - 상태 변환 확률이 1인 벨만 기대 방정식

      $v_\pi(s) = \Sigma_{a\in A}\pi(a|s)(r_{(s,a)} + \gamma v_\pi(s'))$

      → (1) 각 행동에 대해 그 행동을 할 확률을 고려하고

      (2) 각 행동을 했을 때 받을 보상과

      (3) 다음 상태의 가치함수를 고려한다.

  벨만 기대 방정식을 통해 계속 계산하다 보면 언젠가 방정식의 왼쪽 항과 오른쪽 항이 동일해질 것이다.

  → $v_\pi(s)$ 값이 수렴한다.

  → 현재 정책 π에 대한 참 가치함수를 구한 것이다.

  (참 가치함수 : "어떤 정책"을 따라서 움직였을 경우에 받게 되는 보상에 대한 참값)

- **벨만 최적 방정식**

  - 최적 가치함수 : 수많은 정책 중 '가장 높은 보상을 얻게 되는 정책'을 따랐을 때의 가치함수

    : $v_*(s) = max_\pi[v_\pi(s)]$

- 최적 정책 : 모든 정책에 대해 가장 큰 가치함수를 주는 정책

  (선택 상황에서의 판단 기준은 큐함수. 최적 정책은 언제나 이 큐함수 중에서 가장 높은 행동을 하는 것이다.)

  최적 큐함수

  : $q_*(s,a) = max_\pi[q_\pi(s,a)]$

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

순차적 행동 결정 문제를 푸는 것 ⇒ 최적 가치함수 혹은 최적 큐함수를 구하는 것!

벨만 최적 방정식

: $v_*(s) = max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$

큐함수에 대한 벨만 최적 방정식

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

벨만 기대 방정식과 벨만 최적 방정식을 이용해 MDP로 정의되는 문제를 "계산"으로 푸는 방법이 이후에 다룰 다이내믹 프로그래밍이다.