UNIST HeXA 여름 특강

강화학습 기초

Chris Ohk
utilForever@gmail.com

다이내믹 프로그래밍의 한계

정책 이터레이션, 가치 이터레이션은 벨만 방정식을 이용한 다이내믹 프로그래밍이다.

다이내믹 프로그래밍은 계산을 빠르게 하는 것이지, “학습”을 하는 것은 아니다.

정책 이터레이션, 가치 이터레이션으로도 최적 정책을 찾을 수는 있다. (언젠가는)

그런데 우리가 강화학습을 배우는 이유는?

→ 다이내믹 프로그래밍이 여러 한계를 가지고 있기 때문

다이내믹 프로그래밍의 한계

1. 차원의 저주

설명하면서 다루었던 그리드 월드는 상태가 (x, y)로 표현되는 2차원

크기(N)가 5였으니 $5^2 = 25$, 하지만 3차원? 10차원? … ?

차원의 수가 늘어나면 상태의 수가 지수적으로 증가

→ 계산해야 할 양이 기하급수적으로 늘어난다.

다이내믹 프로그래밍의 한계

2. 환경에 대한 완벽한 정보가 필요

정책 이터레이션과 가치 이터레이션을 다룰 때,
환경의 보상과 상태 변환 확률을 정확히 안다는 가정하에 풀었었다.

하지만 보통은 환경에 대한 정보를 정확히 알 수 없기 때문에,
환경과의 상호작용을 통해 경험을 바탕으로 학습하는 방법론이 등장했다.

→ 그것이 바로 강화학습!

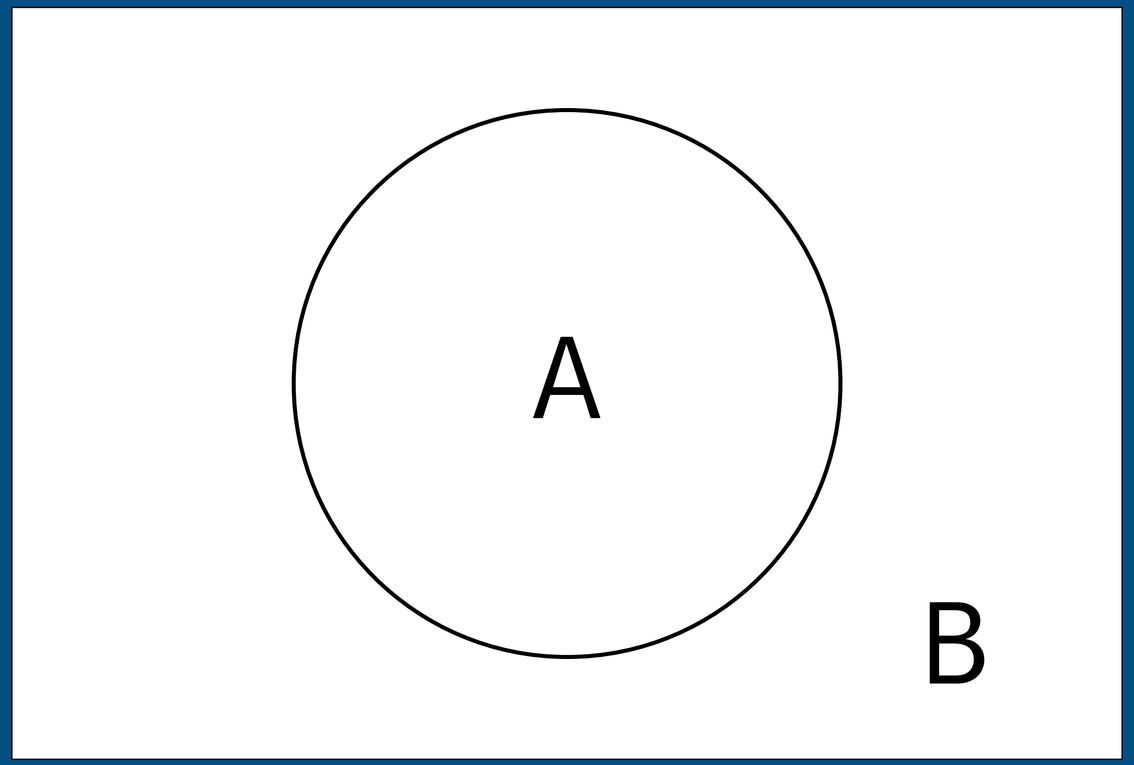
몬테카를로 근사

몬테카를로 : 무작위로 무엇인가를 해본다

근사 : 원래의 값을 추정하는 것

몬테카를로 + 근사 = 무작위로 무엇인가를 해서 원래의 값을 추정한다.

몬테카를로 근사



원의 넓이를 S(A), 직사각형의 넓이를 S(B)라 하면

S(B)와 S(A)/S(B)를 안다면 원의 넓이를 구할 수 있다.

몬테카를로 근사

$\mathrm{S(A)}/\mathrm{S(B)}$를 근사

- 직사각형 위에 무작위로 점을 뿌린다.
- 전체 점 중 원 안에 들어간 점의 개수를 구한다.

무작위로 n개의 점을 뿌렸을 때

$$\frac{\mathrm{S(A)}}{\mathrm{S(B)}} \approx \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\mathrm{dot}_i \in \mathrm{A})$$

$n \to \infty$이면 위 두 수식은 같아진다.

몬테카를로 예측

앞의 점 하나하나는 “샘플”
→ 점을 찍는 것은 “샘플링(Sampling)”

강화학습은 샘플링을 통해 참 가치함수의 값을 근사하는 것
→ 에이전트가 환경에서 에피소드를 진행하는 것이 “샘플링”

몬테카를로 예측 : 몬테카를로 근사를 통해 참 가치함수의 값을 근사

몬테카를로 예측

가치함수 : $v_\pi(s) = E_\pi[G_t|S_t = s]$

→ 에피소드를 통해 반환값을 얻으면 참 가치함수의 값을 근사할 수 있다.

여러 에피소드를 통해 근사한 가치함수

$$v_\pi(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$

- $N(s)$: 여러 에피소드 동안 상태 s에 방문한 횟수
- $G_i(s)$: 상태를 방문한 i번째 에피소드에서 s의 반환값

몬테카를로 예측

V_{n+1} : n개의 반환값을 통해 평균을 취한 가치함수

$$V_{n+1} = \frac{1}{n}\sum_{i=1}^{n} G_i = \frac{1}{n}\left(G_n + \sum_{i=1}^{n-1} G_i \right)$$

$$= V_n + \frac{1}{n}(G_n - V_n)$$

몬테카를로 예측

정리하면, 가치함수의 업데이트 식은

$$V(s) \leftarrow V(s) + \alpha\big(G(s) - V(s)\big)$$

위 수식에서

- $G(s) - V(s)$: 오차
- α : 스탭 사이즈, 오차의 얼마를 가지고 업데이트 할 지를 의미

몬테카를로 예측

몬테카를로 예측에서

- 가치함수를 업데이트하며 도달하려는 목표는 반환값
  → 한 번에 목표점으로 가는 것이 아니고 스탭 사이즈를 곱한 만큼만 간다.
- 에이전트는 에피소드 동안 경험했던 모든 상태에 대해 가치함수를 업데이트 한다.
- 가치함수를 업데이트 하기 위해서는 에피소드가 끝날 때까지 기다려야 한다.
  → 에피소드의 끝이 없거나 에피소드의 길이가 긴 경우에는 적합하지 않다.

시간차 예측

- 실시간으로 가치함수를 업데이트
- 다른 형태의 가치함수를 사용

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

시간차 예측

시간차 예측에서 가치함수의 업데이트

$$V(S_t) \leftarrow V(S_t) + \alpha(R + \gamma V(S_{t+1}) - V(S_t))$$

- 에이전트는 현재의 상태 S_t에서 행동을 하나 선택
- 환경으로 부터 R을 받고 다음 상태 S_{t+1}을 알게 된다.
  → $R + \gamma V(S_{t+1})$을 목표로 가치함수를 업데이트

살사

시간차 제어 = 시간차 예측 + 탐욕 정책

탐욕 정책 : $\pi(s) = \mathrm{argmax}_{a \in A} Q(s,a)$

- 현재 상태의 큐함수를 이용했기 때문에 환경의 모델을 몰라도 된다.
- 업데이트 하는 대상은 큐함수

살사

시간차 제어의 업데이트 수식

$$\mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t) \leftarrow \mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t) + \alpha\left(R_{t+1} + \gamma \mathrm{Q}(\mathrm{S}_{t+1}, \mathrm{A}_{t+1}) - \mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t)\right)$$

여기서 $[\mathrm{S}_t, \mathrm{A}_t, \mathrm{R}_{t+1}, \mathrm{S}_{t+1}, \mathrm{A}_{t+1}]$이 하나의 샘플 → SARSA라고 부름

살사

초기의 에이전트에선 탐욕 정책으로 인해 학습이 잘못될 가능성이 크다.
→ 에이전트에게 충분한 경험을 통해 최적에 가까운 큐함수를 가져야 한다.

대안 : ϵ-탐욕 정책

$$\pi(s) = \begin{cases} a^* = \text{argmax}_{a \in A} Q(s,a), & 1-\epsilon \\ a \neq a^*, & \epsilon \end{cases}$$

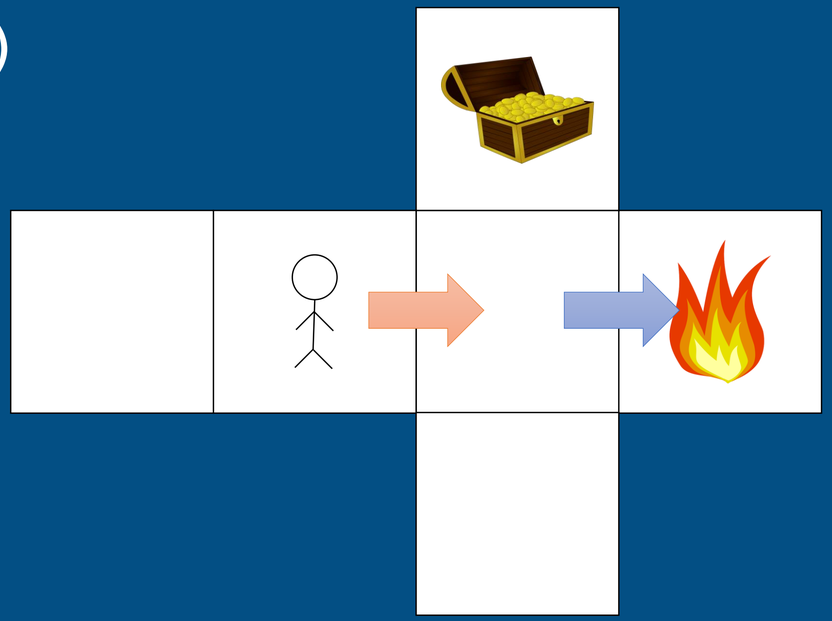
- $1-\epsilon$의 확률로 현재 상태에서 가장 큰 큐함수의 값을 갖는 행동을 선택 = 탐욕 정책
- ϵ의 확률로 엉뚱한 행동을 선택 = 탐험

살사

살사의 문제점 : 살사는 자신이 행동하는 대로 학습하는 시간차 제어
→ 최적 정책을 학습하지 못하고 잘못된 정책을 학습할 수 있음.

예시

- 에이전트가 현재 상황에서 탐욕 정책에 따라 이동(빨간색)
- 그 이후 탐험을 통해 엉뚱한(파란색) 행동을 함
- 빨간색 행동의 큐함수 값이 낮아지게 됨
  → 잘못된 정책을 학습하게 됨



큐러닝

에이전트가 다음 상태 s'을 알게 되면,
그 상태에서 가장 큰 큐함수를 이용해 업데이트
→ 다음 상태에서 어떤 행동을 했는 지와 상관 없이 업데이트

큐러닝을 통한 큐함수 업데이트

$$\mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t) \leftarrow \mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t) + \alpha\left(R_{t+1} + \gamma \max_{\mathrm{a}'} \mathrm{Q}(S_{t+1}, \mathrm{a}') - \mathrm{Q}(\mathrm{S}_t, \mathrm{A}_t)\right)$$

감사합니다!

스터디 듣느라 고생 많았습니다.