




Policy Iteration & Value Iteration

Policy Iteration in RL: An Illustration

Policy Iteration¹ is an algorithm in 'Reinforcement Learning', which helps in learning the optimal policy which maximizes the long term discounted reward. These techniques are often useful, when there are multiple options to chose from, and each option has its own

 <https://towardsdatascience.com/policy-iteration-in-rl-an-illustration-6d58bdc87a7>



서론 : Game Introduction

게임 살펴보기

Reward

본론 1 : Policy Iteration

이론

Policy란 무엇인가?

Step 1 : Initialize Random Policy

Step 2 : Policy Evaluation

Step 3 : Policy Improvement

Step 4 : 반복

Pirate Survival Game

State Transition Diagram

Transition Probability Matrix

Policy Iteration 적용하기

본론 2 : Value Iteration

이론

First Iteration

Second Iteration

s_1

s_2

s_3

Policy 수정 및 Value 다시 계산

Third Iteration부터 생략

결론

서론 : Game Introduction

PI는 강화학습에서 사용되는 알고리즘으로,

장기간의 discounted reward의 관점에서 최적의 policy 선택을 도와준다.

이 방법은 여러개의 policy가 각각의 reward와 risk가 있을 때 유용하다.

이 글에서는 'Policy Iteration' 알고리즘을 간단한 게임에 적용해보고자 한다.

이 게임에서 해적은 목적지로 향하려하는데, 위험한 길과 이로운 길이 있다.

게임 살펴보기

해적선이 어떤 섬에 있으며, 안전하게 집에 가려고 한다. 집에 가는 방법은 2가지가 있다.

1. 북쪽길

북쪽으로 가면 금으로 가득한 섬에서 금을 채울 수 있다.

이 때 남쪽으로 가면 집에 도착하지만, 높은 확률로 다시 북쪽으로 이동하여 침몰한다.(버뮤다 삼각지대)

2. 남쪽길

남쪽으로 가면 은으로 가득한 섬에서 은을 채울 수 있다.

이 때 북쪽으로 이동하여 집에 도착하지만, 다시 남쪽으로 이동하면 수용소 섬에 수감된다.

현재 우리는 “캐리비안 해적”의 Jack Sparrow의 상황처럼, 나침반이 고장났다.

그래서 북쪽으로 이동하면 80%의 확률로 북쪽으로 이동하고, 20%의 확률로 남쪽으로 이동한다고 하자.

대신 남쪽으로 이동했으면, 그 다음에도 남쪽으로 갈 확률이 80%이다.

Reward

배가 도착한 곳에 따라 reward를 주려고 한다.

- 집에 도착 : +1 point
- 금의 섬 도착 : +2 point
- 은의 섬 도착 : +1 point
- 버뮤다 삼각지대 도착 : -2 point
- 수용소 섬 도착 : -0.5point

이러한 상황에서 가장 높은 reward를 가진채로 집에 도착할 수 있는 최적 Policy를 찾고자 한다.

본론 1 : Policy Iteration

Policy Iteration은 아래의 3단계를 거친다.

1. Initialize Random Policy
2. Policy Evaluation
3. Policy Improvement
4. 반복

Policy의 개념과 각각의 단계에 대해서 이야기해보자.

이론

Policy란 무엇인가?

Policy는 우리의 system에서 발생 가능한 모든 state에 대해, action을 연결해준다.

최적 policy는 장기적 관점에서 reward를 최대화 하는 policy이다.

위의 예시에서 많은 수의 policy가 존재할 것이다.

그 중에서도 최고의 보상을 가져다주는 최적의 policy를 하나만 찾아 보자.

Step 1 : Initialize Random Policy

Random하게 policy를 시작하자.

system의 모든 state에서의 action이 random하게 움직인다.

Step 2 : Policy Evaluation

Policy Evaluation은 Bellman 방정식에 의해 다음과 같이 정의된다.

$$V(s) = r(s) + \gamma * \max_{s',r} (\sum p(s',r|s,\pi(s)) * V(s'))$$

특정 policy로 모든 state에 대한 action을 받아와 위의 수식으로 방정식을 풀어낸다.

p 는 전이 확률 행렬이다.

💡 인터넷의 내용과 cs234 Lecture2의 내용이 달라, 강의를 기준으로 다시 정리했다.

$$Q(s) = r(s, a) + \gamma \sum_{s',r} p(s'|s, \pi(s)) V(s')$$

$r(s, a)$: s 에서 a 를 했을 때의 reward

γ : discount factor : 시간이 흐름에 따라 변화

π : policy

$V(s')$: s' 의 Value

$\Rightarrow p(s'|s, \pi(s))$: s 에서 π 에 따라 행동 했을 때 s' 이 나올 전이 확률

$\Rightarrow \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V(s')$: 모든 s 에 대해서, ($\gamma * s'$ 에 대한 전이 확률 * Value)의 합

$\therefore Q(s)$: s 에서 특정 a 를 수행했을 때의 reward와, 그 이후부터 π 를 따라 행동했을 때 얻을 수 있는 Value의 discounted sum의 합

Step 3 : Policy Improvement

모든 state에 대해서, 최고의 Value를 가질 수 있는 행동을 찾는다.

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s', r} p(s', r | s, a) * V(s')$$

a 가 현재의 policy에 따른 action보다 낫다면, best action을 현재의 action으로 바꾼다.

💡 수업에서는 아래의 수식을 이용해서 설명하고, 그 아래는 Inotes에 적혀있다.

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} V^\pi(s)$$

$$\pi^*(s) \leftarrow \underset{a \in A}{\operatorname{argmax}} [R(s, a) + r \sum_{s' \in S} P(s' | s, a) V^\pi(s')], \quad \forall s \in S$$

⇒ 현재 state에서, 가장 좋은 V 를 만들 수 있는 π 를 만들자.

Step 4 : 반복

Step 2와 Step 3가 수렴할 때 까지 반복한다.

policy가 바뀌지 않으면, 알고리즘이 수렴했다고 본다.

Pirate Survival Game

State Transition Diagram

state와 action을 이해하기 위해서는 state transition diagram을 만들어야 한다.

우리 게임에서는 개별 섬이 state이고, “남쪽” or “북쪽” 2개의 action이 있다.

개별 state의 reward는 아까 위에서 정의했다.

⇒ State Transition Diagram을 그려보자.

시작 지점과 도착지점을 포함하여 총 6개의 State가 생긴다.

S1 : 시작지점

S2 : 금의 섬

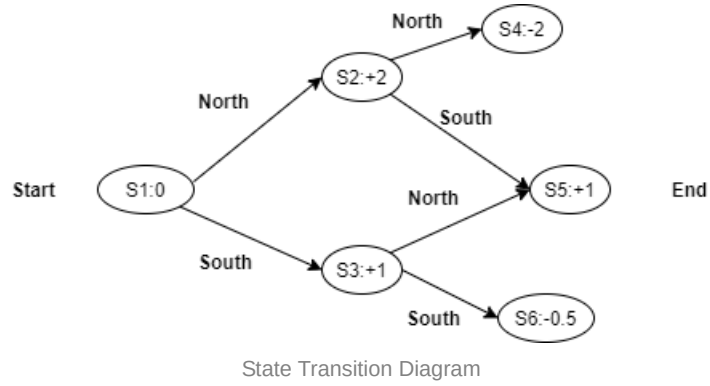
S3 : 은의 섬

S4 : 버뮤다 삼각지대

S5 : 목적지

S6 : 수용소 섬

⇒ Diagram은 아래의 이미지와 같다.



Transition Probability Matrix

현재 가는 방향이 확률적이기 때문에, 개별 state에 따른 전이확률 행렬을 그려야 한다.

위에서 말한 확률에 따르면, Action에 따른 2개의 전이확률 행렬이 나온다.

Bellman 방정식을 적용하면 $P(S, a, S')$ 은 a 를 했을 때 S 에서 S' 으로 갈 확률이다.

$$P[A(\text{North})] = \begin{bmatrix} & S1 & S2 & S3 & S4 & S5 & S6 \\ S1 & 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0.8 & 0.2 & 0 \\ S3 & 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ S4 & 0 & 0 & 0 & 0 & 0 & X \\ S5 & 0 & 0 & 0 & 0 & 0 & X \\ S6 & 0 & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

$$P[A(\text{South})] = \begin{bmatrix} & S1 & S2 & S3 & S4 & S5 & S6 \\ S1 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ S3 & 0 & 0 & 0 & 0 & 0.2 & 0.8 \\ S4 & 0 & 0 & 0 & 0 & 0 & X \\ S5 & 0 & 0 & 0 & 0 & 0 & X \\ S6 & 0 & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

Policy Iteration 적용하기

초기 Policy는 전부 다 북쪽으로 이동한다고 하자.

하지만 s_4, s_5, s_6 에서는 가능한 action이 없다.

따라서 전이 확률 행렬은 s_1, s_2, s_3 에서의 action만 보면 된다.

우선은 γ 는 없이 $Q(s, a)$ 가 아닌 $V(s)$ 의 관점에서 풀어보자.

현재의 최적 Policy를 다음과 같이 세팅하자.

$\{N, N, N\}$

First Iteration

초기 $V(s) = 0$ 으로 설정한다.

Bellman 방정식에 의해 $V(s) = R(s)$ 가 된다.

따라서 첫번째 iteration에서 Policy Evaluation을 적용하면 아래의 Value를 가진다.

$$V(s_1) = 0, \quad V(s_2) = 2, \quad V(s_3) = 1$$

$$V(s_4) = -2, \quad V(s_5) = 1, \quad V(s_6) = -0.5$$

여기에서 Policy Improvement를 구하면, 다음과 같다.

State	Action	V[S]	Max Action
S1	North	$V[S] = 0.8 * 2 + 0.2 * 1 = 1.8$	North
	South	$V[S] = 0.2 * 2 + 0.8 * 1 = 1.2$	
S2	North	$V[S] = 0.8 * -2 + 0.2 * 1 = -1.4$	South
	South	$V[S] = 0.2 * -2 + 0.8 * 1 = 0.4$	
S3	North	$V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7$	North
	South	$V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2$	

First Iteration : Policy Improvement

Policy Improvement에서는 현재 State에 대한 Value를 구한다.

수업 내용과 조금 다른 것 같다...?

수업에서는 $V(s)$ 가 아닌 Discounted Sum이 올라가도록 Policy를 선택한 것 같은데..

우선은 인터넷 자료를 따라서 진행해보자.

위의 예시의 경우, 개별 state에서의 최적의 Policy는

$\{N, N, N\}$ 에서 $\{N, S, N\}$ 으로 바뀌었다.

Second Iteration

먼저 현재의 Policy에 대해서 Evaluation하자.

두 번째의 Evaluation부터는 Discounted Sum으로 Value를 계산한다.

State	Action	V[S]	Total value (Reward + Sum of Value for each Action)
S1	North	$V[S] = (0.8 * 2 + 0.2 * 1) = 1.8$	
	South	$V[S] = (0.2 * 2 + 0.8 * 1) = 1.2$	
			$0 + 1.8 + 1.2 = 3$
S2	North	$V[S] = (0.8 * -2 + 0.2 * 1) = -1.4$	
	South	$V[S] = (0.2 * -2 + 0.8 * 1) = 0.4$	$2 - 1.4 + 0.4 = 1$
S3	North	$V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7$	
	South	$V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2$	$1 + 0.7 - 0.2 = 1.5$

Second Iteration : Policy Evaluation

따라서 각각의 Value는 다음과 같다.

$$V(s_1) = 3, \quad V(s_2) = 1, \quad V(s_3) = 1.5$$

$$V(s_4) = -2, \quad V(s_5) = 1, \quad V(s_6) = -0.5$$

이를 바탕으로 Policy Improvement를 하면 아래와 같다.

State	Action	V[S]	Max Action
S1	North	$V[S] = 0.8 * 1 + 0.2 * 1.5 = 1.1$	
	South	$V[S] = 0.2 * 1 + 0.8 * 1.5 = 1.4$	
			South
S2	North	$V[S] = 0.8 * -2 + 0.2 * 1 = -1.4$	
	South	$V[S] = 0.2 * -2 + 0.8 * 1 = 0.4$	South
S3	North	$V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7$	North
	South	$V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2$	

Second Iteration : Policy Improvement

Second Iteration의 Policy Evaluation에서 구한 것을 바탕으로,
개별 State에 대한 Value가 바뀌었다.

s_1 의 경우 North는 (1.8 → 1.1)로, South는 (1.1 → 1.4)로 바뀌면서
최적의 Action이 South가 되었다.

이에 따른 최적의 Policy는 $\{S, S, N\}$ 이 되었다.

Thrid Iteration(자세한 설명 생략)

Policy Evaluation

State	Action	V[S]	Total value (Reward +Sum of Value for each Action)
S1	North	$V[S] = (0.8 * 1 + 0.2 * 1.5) = 1.1$	
	South	$V[S] = (0.2 * 1 + 0.8 * 1.5) = 1.4$	
			$0 + 1.1 + 1.4 = 2.5$
S2	North	$V[S] = (0.8 * -2 + 0.2 * 1) = -1.4$	
	South	$V[S] = (0.2 * -2 + 0.8 * 1) = 0.4$	$2 - 1.4 + 0.4 = 1$
S3	North	$V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7$	
	South	$V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2$	$1 + 0.7 - 0.2 = 1.5$

Third Iteration : Policy Evaluation

State	Action	V[S]	Max Action
S1	North	$V[S] = 0.8 * 1 + 0.2 * 1.5 = 1.1$	
	South	$V[S] = 0.2 * 1 + 0.8 * 1.5 = 1.4$	
			South
S2	North	$V[S] = 0.8 * -2 + 0.2 * 1 = -1.4$	
	South	$V[S] = 0.2 * -2 + 0.8 * 1 = 0.4$	South
S3	North	$V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7$	North
	South	$V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2$	

Third Iteration : Policy Improvement

최적의 Policy : $\{S, S, N\}$

⇒ 최적의 Policy가 바뀌지 않았다.

따라서 Policy가 수렴했다고 판단, Iteration을 중단하고, 최적 Policy로 사용한다.

본론 2 : Value Iteration

이론

너무 자세한 내용은 위에서 설명했기에 생략하자.

$V(s)$ 를 0으로 초기화하고,
 $k = 0, A = \{N, N, N\}, \gamma = 0.5$ 를 수행한다고 하자.

기본적인 $V(s)$ 는 아래의 식을 만족한다.

$$V(s) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(s')$$

Value Iteration에서는 $V(s)$ 를 조정해 $R(s, a)$ 를 최대화 하도록 조정한다.

즉, 현재 상태에서의 최선의 action을 수행하면, 최종 Value가 증가 할 것이란 관점이다.

따라서 다음의 수식을 만족하도록 각각의 s 에서의 a 를 고르자.

$$V(s) = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(s')$$

First Iteration

첫 Iteration에서는 단순히 $V(s) = R(s)$ 를 계산해주자.

$$V(s_1) = 0, \quad V(s_2) = 2, \quad V(s_3) = 1 \\ V(s_4) = -2, \quad V(s_5) = 1, \quad V(s_6) = -0.5$$

Second Iteration

s_1

- $a = \{N\}$:

$$\begin{aligned} V(s_1) &= R(s_1, a(North)) + \frac{1}{2} (p_N(s_2|s_1, a(North))V(s_2) + p_N(s_3|s_1, a(South))V(s_3)) \\ &= 2 + \frac{1}{2} (0.8 * 2 + 0.2 * 1) = 2 + (0.4 * 2 + 0.1 * 1) = 2.9 \end{aligned}$$

해석 :

$R(s_1, a(North))$: s_1 에서 a 가 North 일 때의 Reward = 2

$p_N(s_2|s_1, a(North))V(s_2)$

- p_N : 내 action이 north일 때의 probability

- $p_N(s_2|s_1, a(North))$: s_1 에서 $a(North)$ 를 수행해 s_2 로 이동할 확률

- $p_N(s_2|s_1, a(North))V(s_2)$: s_1 에서 s_2 로 이동했을 때의 value

- $a = \{S\}$:

$$V(s_1)$$

$$\begin{aligned}
&= R(s_1, a(South)) + \frac{1}{2} (p_S(s_2|s_1, a(North))V(s_2) + p_S(s_3|s_1, a(South))V(s_3)) \\
&= 1 + \frac{1}{2}(0.2 * 2 + 0.8 * 1) = 1 + (0.1 * 2 + 0.4 * 1) = 1.6
\end{aligned}$$

$\Rightarrow s_1$ 에서는 {N} 선택

s_2

- $a = \{N\}$:

$$\begin{aligned}
&V(s_2) \\
&= R(s_2, a(North)) + \frac{1}{2} (p_N(s_4|s_2, a(North))V(s_4) + p_N(s_5|s_2, a(North))V(s_5)) \\
&= -2 + \frac{1}{2}(0.8 * (-2) + 0.2 * 1) = -2 + (0.4 * (-2) + 0.1 * 1) = -2.7
\end{aligned}$$

- $a = \{S\}$:

$$\begin{aligned}
&V(s_2) \\
&= R(s_2, a(South)) + \frac{1}{2} (p_S(s_4|s_2, a(North))V(s_4) + p_S(s_5|s_2, a(South))V(s_5)) \\
&= 1 + \frac{1}{2}(0.2 * (-2) + 0.8 * 1) = 1 + (0.1 * (-2) + 0.4 * 1) = 1.2
\end{aligned}$$

$\Rightarrow s_2$ 에서는 {S} 선택

s_3

- $a = \{N\}$:

$$\begin{aligned}
&V(s_3) \\
&= R(s_3, a(North)) + \frac{1}{2} (p_S(s_5|s_3, a(North))V(s_5) + p_S(s_6|s_3, a(South))V(s_6)) \\
&= 1 + \frac{1}{2}(0.8 * 1 + 0.2 * (-0.5)) = 1 + (0.4 - 0.05) = 1.35
\end{aligned}$$

- $a = \{S\}$:

$$\begin{aligned}
&V(s_3) \\
&= R(s_3, a(South)) + \frac{1}{2} (p_S(s_5|s_3, a(North))V(s_5) + p_S(s_6|s_3, a(South))V(s_6)) \\
&= 1 + \frac{1}{2}(0.2 * 1 + 0.8 * (-0.5)) = 1 + (0.1 - 0.2) = 0.9
\end{aligned}$$

$\Rightarrow s_3$ 에서는 {N} 선택

Policy 수정 및 Value 다시 계산

$$A = \{N, N, N\} \Rightarrow A = \{N, S, N\}$$

$$V(s_1) = R(s_1) + V(s_1) = 0 + (2.9 + 1.6) = 4.5$$

$$V(s_2) = R(s_2) + V(s_2) = 2 + (-2.7 + 1.2) = 0.5$$

$$V(s_3) = R(s_3) + V(s_3) = 1 + (1.35 + 0.9) = 3.25$$

Third Iteration부터 생략

⇒ 대충 봐도 이번 Iteration에서는 s_1 에서 s_3 로 가기 위해 South로 갈 거 같다.

⇒ $\{S, S, N\}$ 이 되겠다.

네번째 Iteration에서는 Third에서 구한 Value와 Fourth에서 구한 Value가 비슷해, 안정화될 것이다.

⇒ 그러면 Iteration을 중단하고, 최적의 Policy로 선택한다.

결론

최적의 Policy 획득!