Model-based

- Dynamic Programming

Model-free

- Monte carlo method
- Temporal-Difference Learing
  - Sarsa
  - Q-learing

- Dynamic Programming

정책을 결정하기 위해서는 가치 함수를 계산하는 것이 필요하다. 상태 변화의 확률 값 P와 보상R을 미리 알고 있다면 동적 계획법으로 가치 함수를 효율적으로 결정할 수 있다. 작은 문제 -> 큰 문제 해결 방식으로 1.정책 반복 방법과 2.가치 반복 방법이 있다.

정책 반복

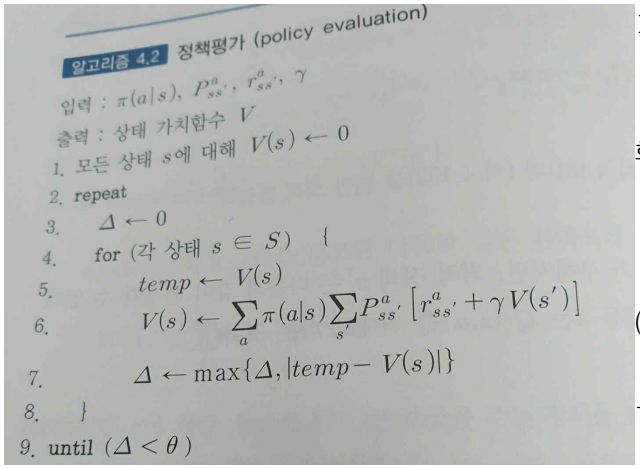
1) 임의의 정책 Π에서 시작해서 Π에 대해서 벨만 기대 방정식

MAX $\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r+\gamma v_\pi(s')\right]$, (벨만 방정식의 파라미터 a에 대한 최대값)에 수렴할 때 까지 계산을 먼저 한다 (정책 평가 과정)

2) V를 사용하여 정책 Π를 개선하는 과정(정책 개선)

1)과 2)를 반복하여 정책Π가 수렴할 때까지 반복하여 최적의 정책Π를 개산한다.

1) 정책 평가 과정 알고리즘



1. 기존의 Policy를 따라 state-value를 계산하자

-먼저 모든 state를 V(s) = 0 으로 초기화 시킨다

-기존 정책 b에서 시작

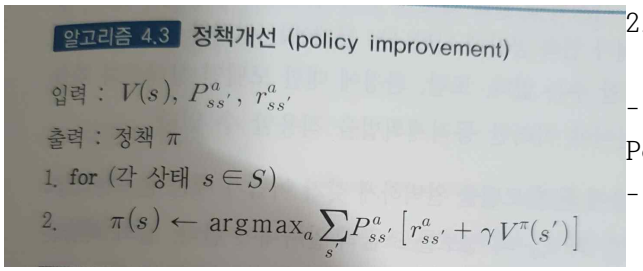
-V(s) = Policy * 전이 확률(P) * (Reward + next state estimated value)

-업데이트하며 V(s)의 변화량이 매우 작을 때 업데이트를 멈춘다

Policy를 따라 Value function을 계산한 이유는 더 나은 Policy를 찾기 위해서 이다

이제 Iteration을 반복하며 true value func.을 찾았으면 이 policy를 따르는 것이 좋은지 안좋은지를 판단하고 Policy를 update 해야 한다.

2) 정책 개선 과정 알고리즘



2. 더 좋은 Policy를 찾자

- 상태 가치 함수 V(s)로부터 보다 나은 Policy 정책을 찾는다.
- 가치가 가장 높은 state를 가겠다, 즉 max값만을 선택을 한다.

동적계획법의 한계

- 모든 상태에 대해서 가치함수 값을 결정해야 하는 table 형태로 상태에 대한 정보가 주어져야 한다. 하지만 현실은 연속된 상태가 많고, 연속된 공간에서 정책 반복이나 가치 반복 등의 동적계획법의 실행이 불가능하다. 그렇기에 강화학습에서는 에이전트가 환경과 상호작용을 하며 P와 r의 정보를 알아 내야 한다. 환경과 상호 작용을 하며 경험을 통해 최적의 정책을 찾는 방법이 Model-free 기법이다.

Model-free

강화 학습에서 예측이란

환경 모델이 없는 상태에서 환경과 상호작용하면서 주어진 정책(Π)에 대한 가치함수를 구하는 것(정책평가를 시행하는 것)

- Monte carlo method
- Temporal-Difference Learing

강화 학습에서 제어란

가치함수를 통해 정책을 개선해 가는 것을 제어라고 한다.

- Sarsa
- Q-learing

- Monte carlo method

Monte Carlo는 episode-by-episode로 업데이트 한다. 에피소드의 마지막 state까지 가서 업데이트 한다. Monte Carlo는 경험을 하며 return 된 sample을 이용하여 state-action value를 평균하여 업데이트한다. 즉 상태변화 확률 P와 보상r 등 환경에 대한 모델 정보가 없어 가치 함수를 계산 못 할 때, 하나의 샘플들을 만들어가면서 점진적으로 가치 함수를 갱신하는 기법이다.

V(s + 1) <- V(s) + α(G(s) - V(s))

V(s) : 상태s의 현재 가치함수 값

G(s) : 하나의 에피소드가 끝난 후 상태s에 대한 누적 보상값

α : 학습률 (<1, 양수값)

- Temporal-Difference Learing

V(s + 1) <- V(s) + α(r+ γ V(s')- V(s))

Monte Carlo 처럼 모델 없이 경험을 통하여 value를 측정하며 DP처럼 끝까지 가지 않아도 중간중간 value를 estimate하는것이 가능한 기법이다.

α(r+ γ V(s')- V(s)) <- 시간차 오류라고 한다. 시간차 예측은 에이전트가 다음 상태s'에 대해 현재의 가치함수 값V(s')을 사용하여 현재 상태의 가치함수 값 V(s)를 예측하는데 자신의 정보를 이용한다. [**bootstrap**]

- Monte carlo method
- Temporal-Difference Learing

이 두 가지 방식은 모두 on-policy control이다. 현재 움직이고 있는(action을 취하는) 정책과 impove하는 정책[정책 개선 과정 알고리즘]이 같다. on-policy 는 gredy 방식으로 정책을 향상시키는데 한계가 있다.

그러므로 1)ε-greedy improvement이나

2)off-policy 방식 [action을 취하는 policy 과 improve policy를 다르게 설정]

위 두 가지 방식으로 정책을 개선 과정 알고리즘을 향상 시킬 수 있다. 이유 : exploration을 계속하면서도 optimal한 policy를 학습할 수 있다.

강화학습에서의 행동은 Discrete action과 Continuous action으로 나눌 수 있다. 알고리즘에 따라 특정 action space는 학습이 아예 불가능한 경우도 있기 때문에 강화학습 알고리즘을 선정할 때 중요한 요인 중 하나이다.

Discrete action

Discrete 하게 결정되는 행동을 말한다. 즉, 중간 값이 없고 0 또는 1의 값을 갖습니다.

예를 들어, [전진, 후진, 회전] 이라는 행동이 있을 때 전진 행동은 [1, 0, 0]과 같이 선택합니다.중간 값은 없으며 Discrete action으로 다양한 행동을 표현하려면 행동의 개수를 늘려야합니다.

예시: [전진 1cm, 전진 2cm, 후진 1cm, 후진 2cm, 회전]

주로 Q learning과 같은 value-based 알고리즘으로 학습합니다.

Continuous action

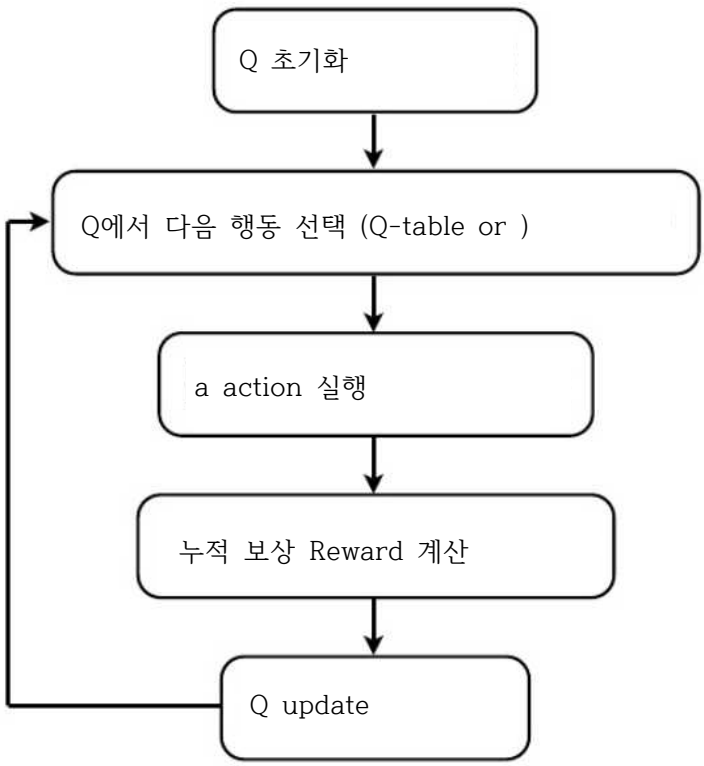
Continuous action은 실수 값을 가지며 값의 크기를 표현할 수 있습니다.

예를 들어, [전/후진, 회전] 이라는 행동이 있을 때 [0.3, 0.4]과 같이 전/후진, 회전 값을 실수 값으로 지정할 수 있습니다.

행동 값의 범위가 무한하기 때문에 action space의 크기가 큽니다.

주로 Policy gradient와 같은 policy-based 알고리즘으로 학습합니다.

off-policy + discrete action = DQN



DQN이 나온 이유

1. Q-Network의 **2가지 문제점을** 해결하기 위해 3가지 방법을 제시

   Q-Network의 문제점

   1) Sample Data들의 연관성

      State에서 Action을 취한 뒤 Reward를 얻은 다음 받은 State는 이전 State와 굉장히 유사하다 Q-Learning 알고리즘에서 '연속된 유사한 State를 사용해서 학습을 진행하면 학습 방향이 이상한 방향으로 튄다' Sample Data들이 유사하기 때문에 연관성도 고려 해줘야 한다

   2) 비교해야 하는 값의 불안전성

      현재 State에서 Q값을 비교해야 할 Target값에서 사용되는 다음 State에서 Q값은 같은 θ를 사용한다. -> 동일한 Neural Network를 사용해서 학습한다 와 동일한 말

      Neural Network는 예측값과 Target값의 차이의 제곱을 더하는 cost 함수를 반복 학습하면서 오차를 줄여나가는 것이다. 학습을 하면서 바뀐 θ가 다음 State의 값에도 영향을 끼치게 되면서 예측값과 비교해야 할 Target값 또한 바뀐다. 결국 비교 대상이 지속적으로 영향을 받아 바뀌게 되어 학습이 잘 안된다.

DQN이 제시한 해결 방안

1. Neural Network의 층을 여러개 만들어 deep learning 시킨다.
2. Experience Replay 버퍼에 저장해서 경험했던 것을 가져와 쓴다.
3. Separate Networks 네트워크를 분리한다.

1. CNN에서도 쓰이는 Neural Network에서 학습율을 높이는 일반적인 방식이다,

2. Q-network의 문제점인 Sample Data 간의 연관성 문제를 해결하는 방식이다. 간단히 설명하면 State에서 선택 가능한 모든 Action들에 대해서 Reward와 다음 State에 대한 정보를 모두 Buffer에 저장한 후 Random하게 Data를 선택하여 sample data를 만드는 방식이다. 여러곳(local)의 data를 랜덤하게 취득한 후 학습을 시키면 결과들을 모두 수렴하게 하여 가장 좋은 하나의 linear 곡선을 생성할 수 있다.

3. 네트워크를 분리하는 방법은 Target값의 불안전성을 해결하는 방법이다. Q-Network의 수식에서 같은 θ를 쓰는 것을 θ ,θ‘를 가지는 각각의 Network를 구성하여 문제를 해결한다. 먼저 State에서 사용되는 1번째 Neural Network만을 변화시켜서 학습을 해 준 다음,
다음 State에서 사용되는 2번째 Neural Network를 현재 state에서 사용되는 것과 동일하게 바꾸어 준다. 이렇게 하면 Target의 불안정성을 해결할 수 있다.

출처)
책 인공지능: 튜링 테스트에서 딥러닝까지 (저자: 이건명)
블로그 https://sumniya.tistory.com/10
pdf 강화학습의 이론과 실제 (모두의 연구소 강사 정원석)