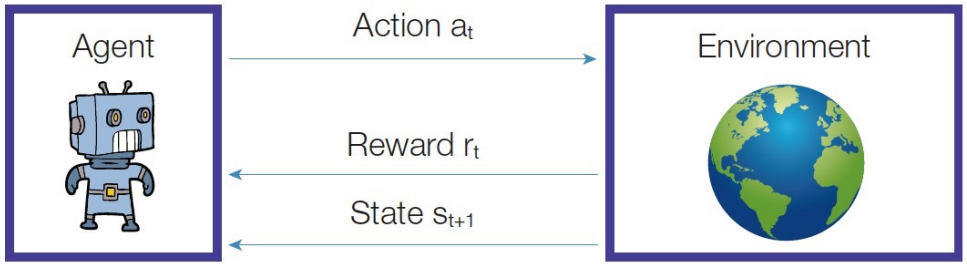
PyTorch로 강화학습 에이전트 만들기 - DQN

오늘의 학습내용

- 강화 학습의 기본 개념과 MDP Markov Decision Process
- 상태 가치 함수 State-Value Function
- 행동 가치 함수 Action-Value Function
- DQN Deep-Q-Networks
- 리플레이 메모리 Replay Memory
- ϵ-Greedy Method
- PyTorch를 이용한 DQN 에이전트 구현

강화 학습 (Reinforcement Learning)

- 강화 학습Reinforcement Learning은 앞서 살펴본 지도 학습과 비지도 학습과는 학습하는 방법이 조금 다른 기법입니다.
- 앞서 살펴본 알고리즘들은 데이터가 이미 주어진 정적인 상태Static Environment에서 학습을 진행했다면, 강화 학습은 에이전트Agent가 주어진 환경State에서 어떤 행동Action을 취하고 이에 대한 보상Reward을 얻으면서 학습을 진행합니다.



Reinforcement Learning Setup

그림 1-7 | 강화 학습의 기본 구성 요소

강화 학습 (Reinforcement Learning)

- 이때 에이전트는 보상Reward을 최대화Maximize하도록 학습을 진행합니다.
- 즉, 강화 학습은 동적인 상태Dynamic Environment에서 데이터를 수집하는 과정까지 학습 과정에 포함되어 있는 알고리즘입니다.

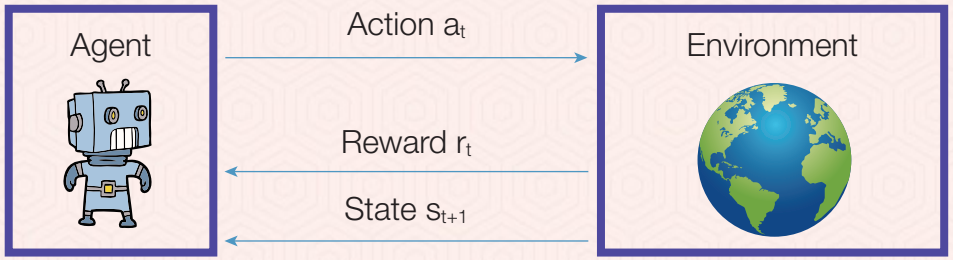
강화 학습의 기본 개념과 MDP Markov Decision Process

- 이번 시간에는 지도 학습, 비지도 학습과 함께 대표적인 머신러닝 방법론 중 하나인 강화 학습Reinforcement Learning에 대해 알아봅시다. 지금까지 배운 학습 방법들은 데이터가 주어진 상태에서 일부 데이터는 트레이닝에 사용하고, 트레이닝이 끝나면 테스트 데이터에 대해서 예측을 수행하는 형태였습니다.

- 하지만 강화 학습에서는 데이터가 주어지는 것이 아니라 에이전트Agent와 환경Environment이 주어집니다. 데이터는 에이전트가 주어진 환경에서 어떤 행동을 하면서 직접 수집합니다. 좀 더 엄밀하게 말해서 에이전트는 현재 상태State s에서 어떤 행동Action a을 취하고 이에 대한 보상Reward r을 얻습니다. 지금까지 설명한 요소들이 강화 학습의 기본 구성 요소입니다.

강화 학습의 기본 개념과 MDP Markov Decision Process

- 강화 학습의 기본 구성 요소를 좀 더 엄밀하게 나타내면 다음 식과 같이 정리할 수 있습니다. 이런 형태의 모델링을 좀 더 전문적인 용어로 MDP Markov Decision Process라고 합니다.
- 아래 그림은 강화 학습의 구성 요소들을 그림으로 나타냅니다.



Reinforcement Learning Setup

강화 학습의 기본 개념과 MDP Markov Decision Process

- 예를 들어서, 그림과 같은 Atari Breakout 게임을 생각해봅시다. 여기서 에이전트는 벽돌깨기를 플레이하는 막대로 볼 수 있습니다. 또, 현재 상태 s는 벽돌깨기 게임의 현재 픽셀 화면(예를 들면, 80×160×3 차원의 RGB 이미지)으로 생각할 수 있습니다. 현재 상태에서 에이전트가 취할 수 있는 행동 a은 [좌로 움직이기, 가만히 있기, 우로 움직이기] 3가지입니다.

- 에이전트는 현재 상태에서 취한 행동에 따라 만약 공을 제대로 쳐서 벽돌을 많이 깨뜨렸다면 큰 보상, 만약 공을 떨어뜨려서 생명을 잃었다면 작은 보상 R(s, s′)을 얻게 됩니다. 에이전트의 행동에 따라 현재 상태 s(현재 화면)는 각기 다른 다음 상태 s′(다음 화면)로 넘어가게 됩니다. 이런 과정이 무수히 반복되면서 에이전트는 계속해서 $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}(\boldsymbol{s}, \boldsymbol{s}'), \boldsymbol{s}')$ 형태의 데이터를 수집해나갑니다.

강화 학습의 기본 개념과 MDP Markov Decision Process

- 이제 강화 학습의 기본적인 이론적 토대를 살펴봅시다. 강화 학습은 보상 가정 Reward Hypothesis을 기반으로 학습을 진행합니다. 보상 가정 Reward Hypothesis은 모든 목표는 보상을 최대화하는 형태로 나타낼 수 있다는 가정입니다.

- 상식적으로 어떤 상황에서 보상이 최대화되는 행동을 계속하는 것이 의사결정 과정에서 취할 수 있는 가장 최적의 전략이 될 것입니다. 즉, 강화 학습은 보상이 최대화되는 전략을 찾는 것을 목표로 합니다. 좀더 엄밀한 용어로 강화 학습에서는 현재 상태에서 어떻게 행동할 것인지를 결정하는 전략을 정책 Policy π로 표현하고 최적의 정책 π^*를 찾는 것을 학습의 목표로 합니다.

- 그렇다면 어떻게 최적의 정책 π^*을 찾을 수 있을까요? 이를 찾기 위해서 강화 학습은 상태 가치 함수 State-Value Function와 행동 가치 함수 Action-Value Function라는 개념을 사용합니다.

상태 가치 함수State-Value Function

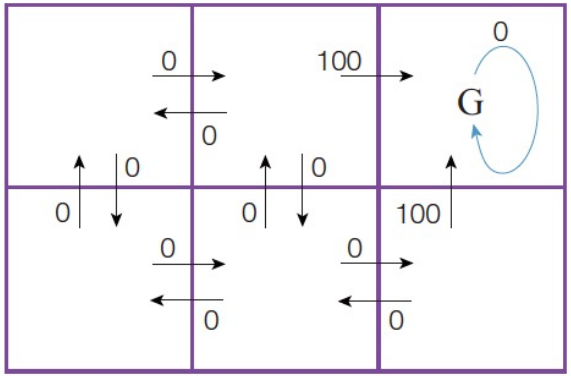
❖ 상태 가치 함수State-Value Function는 현재 상태의 좋음과 나쁨을 표현합니다. 상태 가치 함수를 수학적으로 표현하면 아래와 같습니다.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s]$$

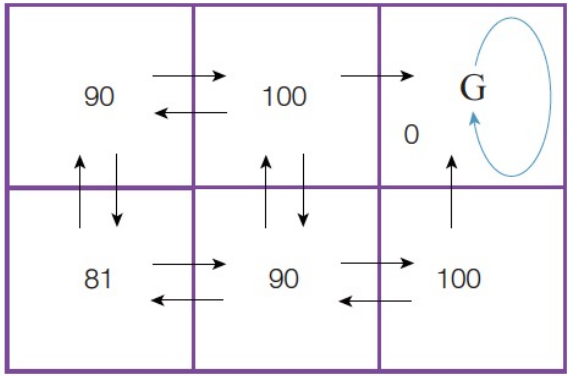
❖ 위의 식이 의미하는 바를 해석해봅시다. 어떤 시간 t에서 전략 π를 따를 때 기대되는 어떤 상태 의 가치는 미래의 보상들의 총합의 평균 으로 표현됩니다. 여기서 γ*는 Discount Factor이고 γ는 보통 0에서 1의 값을 부여합니다. 만약 γ가 0이라면 에이전트는 오직 다음시간(t+1)의 보상만을 고려합니다. 이때의 장점은 빨리 최적의 행동을 결정할 수 있다는 점이고, γ가 1이라면 미래의 보상도 바로 다음의 보상만큼 중요하게 생각합니다. 이 경우, 당장의 보상은 최대화 할 수 없지만 미래의 수까지 내다보면서 행동을 할 수 있다는 장점이 있습니다. 실제 상황에서는 문제에 따라 최적의 γ이 다르고, 실험을 통해 최적의 γ을 설정해주어야 합니다.

❖ 정리하자면, 상태 가치 함수 는 현재 상태 s의 가치를 정량적으로 나타내줍니다. 다음 그림은 G(=Goal)에 도달하면 미로를 탈출해서 100의 보상Reward r을 얻는 그리드 월드Grid World에서 각각의 위치가 갖는 상태 가치 함수값을 보여줍니다. G 지점에 가까울수록 미로를 탈출할 확률이 높기 때문에 높은 상태 가치 함수값, G 지점에서 멀수록 미로를 탈출할 확률이 낮기 때문에 낮은 상태 가치 함수값을 가지고 있는 것을 볼 수 있습니다

상태 가치 함수State-Value Function



$r(s, a)$ (immediate reward)values



$V^*(s)$ values

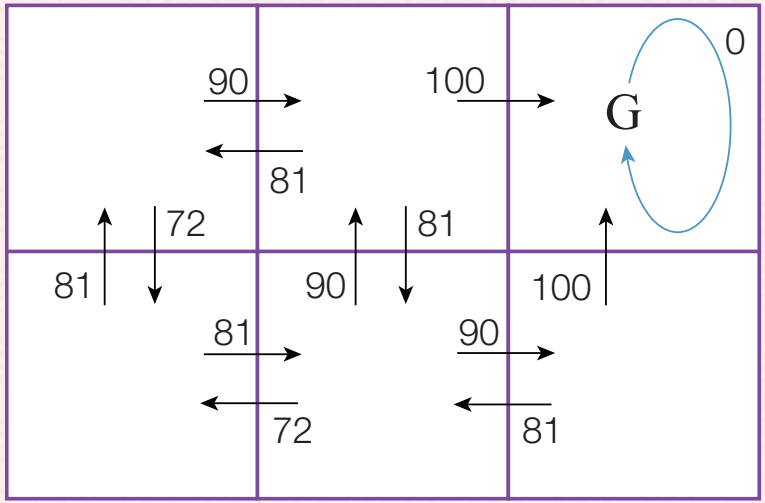
행동 가치 함수Action-Value Function

- 행동 가치 함수Action-Value Function는 현재 행동의 좋음과 나쁨을 표현합니다. 행동 가치 함수를 수학적으로 표현하면 아래와 같습니다.

$$Q_\pi(s,a) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s, A_t = a]$$

- 위의 식이 의미하는 바를 해석해봅시다. 어떤 상태 에서 전략 π를 따를 때 기대되는 어떤 행동 의 가치는 미래의 보상들의 총합의 평균으로 표현됩니다. 상태 가치 함수와 마찬가지로 γ는 현재 보상과 미래 보상의 중요도 가중치를 나타내는 Discount Factor입니다. 정리하자면, 행동 가치 함수 는 현재 상태 s에서 한 행동 a의 가치를 정량적으로 나타내줍니다.
- 다음 그림은 이전 그림에서 살펴본 그리드 월드 상황에서 각각의 위치에서 한 행동의 행동 가치 함수를 나타냅니다. 미로를 탈출하는 G로 이동하는 행동은 가장 큰 행동 가치 함수값인 100, G에서 멀어지는 행동은 낮은 행동 가치 함수값을 갖는 것을 알 수 있습니다.

행동 가치 함수Action-Value Function



$Q(s, a)$ values

행동 가치 함수Action-Value Function

- 행동 가치 함수Action-Value Function는 현재 행동의 좋음과 나쁨을 표현합니다. 행동 가치 함수를 수학적으로 표현하면 아래와 같습니다.

$$Q_\pi(s,a) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s, A_t = a]$$

- 위의 식이 의미하는 바를 해석해봅시다. 어떤 상태 에서 전략 π를 따를 때 기대되는 어떤 행동 의 가치는 미래의 보상들의 총합의 평균으로 표현됩니다. 상태 가치 함수와 마찬가지로 γ는 현재 보상과 미래 보상의 중요도 가중치를 나타내는 Discount Factor입니다. 정리하자면, 행동 가치 함수 는 현재 상태 s에서 한 행동 a의 가치를 정량적으로 나타내줍니다.
- 다음 그림은 이전 그림에서 살펴본 그리드 월드 상황에서 각각의 위치에서 한 행동의 행동 가치 함수를 나타냅니다. 미로를 탈출하는 G로 이동하는 행동은 가장 큰 행동 가치 함수값인 100, G에서 멀어지는 행동은 낮은 행동 가치 함수값을 갖는 것을 알 수 있습니다.

행동 가치 함수Action-Value Function

- 상태 가치 함수와 행동 가치 함수를 통해서 알아내고자 하는 우리의 최종 목표는 최적의 정책 π*를 찾는 것입니다. 어떤 환경에서 모든 상태의 상태 가치 함수값을 알아낸다면 항상 상태 가치가 최대화되는 방향으로 이동하는 것이 최적의 전략이 될 것입니다. 반대로 어떤 환경에서 모든 행동 가치 함수값을 알아낸다면 항상 행동 가치 함수가 최대화 되는 방향으로 행동하는 것이 최적의 전략이 될 것입니다.

- 이렇게 보상이 최대화되는 최적의 정책을 알아내는 방법은 크게 2가지가 있습니다. 상태 가치 함수를 이용해서 알아내는 방법을 Planning이라고 하고 행동 가치 함수를 이용해서 알아내는 방법을 강화 학습Reinforcement Learning이라고 합니다. Planning을 이용하기 위해서는 환경에 대한 모델Model 정보를 알고 있어야 합니다. 좀 더 엄밀히 말하면 에서 Transition Probability P(s,s')에 대한 정보와 이에 대한 보상값 R(s,s')을 알고 있어야 합니다. 행동 가치 함수를 이용한 강화 학습 방법은 모델에 대한 정보가 없더라도 에이전트가 주어진 환경에서 행동을 취하고 얻은 실제 경험Experience을 통해 학습을 진행할 수 있습니다.

- 강화 학습에서 적절한 행동 가치 함수값을 알아내기 위한 구체적인 알고리즘은 SARSA, Q-Learning, Policy Gradient 등이 있습니다.

DQN Deep-Q-Networks

- DQN Deep-Q-Networks은 알파고로 유명한 딥마인드 Deepmind사에서 창안한 강화 학습을 위한 인공신경망 구조로써 기존에 제안된 Q-Networks와 딥러닝 Deep Learning을 결합한 기법입니다.
- DQN은 사전 지식이 없이 오직 게임 화면만을 보고 학습을 진행해서 49개의 아타리 게임 중 24개의 게임에 대해서 인간보다 뛰어난 능력을 보여주어서 많은 사람들을 놀라게 하였습니다.
- 구체적으로 DQN은 기존의 Q-Networks에서 2가지를 개선하였습니다.
- 기존의 Q-Networks은 얕은 층의 ANN 구조를 사용했지만 DQN은 깊은 층의 CNN 구조를 사용했습니다.
- 인간의 해마 Hipocampus에서 영감을 받아 리플레이 메모리 Replay Memory라는 기법을 사용했고 이를 이용해서 효율적인 강화 학습이 가능하도록 만들었습니다.

리플레이 메모리Replay Memory

- 리플레이 메모리Replay Memory 기법에 대해서 자세히 설명하면, 기존의 강화 학습 기법은 에이전트가 획득하는 순차적인 경험데이터 을 이용해서 학습을 진행했습니다. 하지만 이는 에이전트가 취한 이전 행동에 따라 수집되는 데이터의 형태가 일정한 패턴으로 고정되는 문제점이 있습니다.

- 리플레이 메모리 기법은 이를 방지하기 위해서 리플레이 메모리라는 일종의 경험 저장소에 에이전트가 수집한 경험 데이터 를 저장합니다. 리플레이 메모리 기법은 리플레이 메모리에 저장된 경험 데이터에서 랜덤 샘플링을 통해 데이터를 추출해서 이를 학습에 사용함으로써 에이전트가 특정한 패턴에 대한 편견없이 학습할 수 있도록 도와줍니다.

Exploration and Exploitation Trade-off

- Exploration : 학습을 위해 수집하는 Sample의 다양성을 위해서 모험적인 Action을 취하는 것 = 미래의 이익을 최대화
- Exploitation : 최적의 Action을 취하는 것 = 현재의 이익을 최대화

→

- 학습과정에서 적절한 비율의 Exploration과 Exploitation을 취해야 한다.

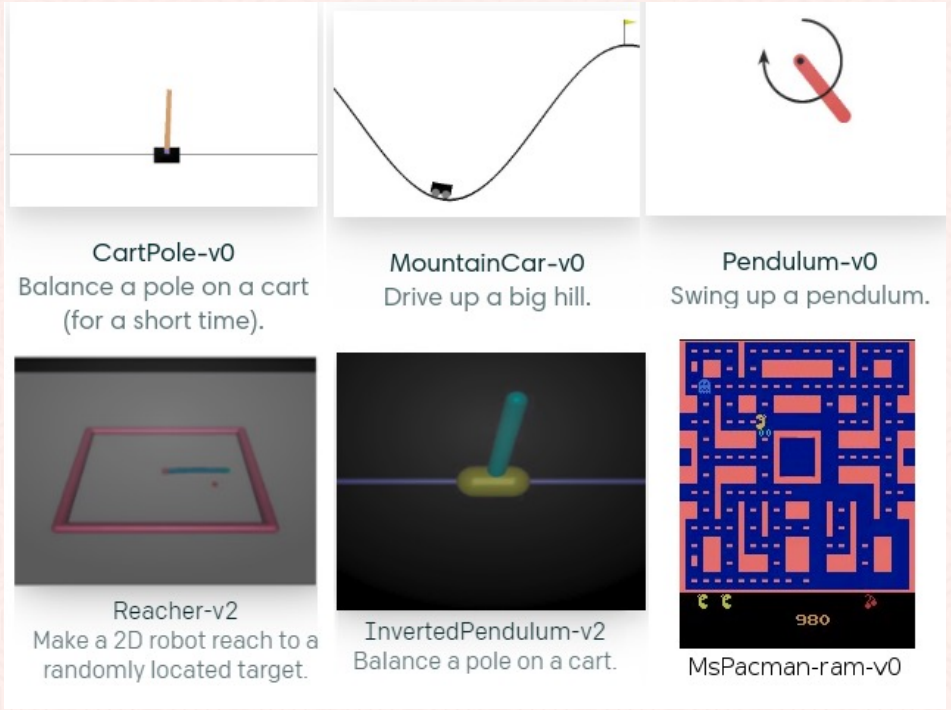
ϵ-Greedy Method

❖ ϵ-Greedy Method : Exploration and Exploitation Trade-off를 해결하기 위한 한가지 방법론 중에 하나

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

OpenAI Gym

- https://gym.openai.com/



OpenAI Gym - CartPole

- https://github.com/openai/gym/wiki/CartPole-v0
- Observation : [Cart Position(카트의 위치), Cart Velocity(카트의 속도), Pole Angle(막대기의 각도), Pole Velocity At Tip(막대기 끝의 속도)]
- Reward : 넘어지지 않을 경우 매 시간마다 +1의 값을 갖는다.
- Action Space : Agent가 취할 수 있는 행동(action_space)은 0, 1 두개의 값이다. 0은 왼쪽으로 이동, 1은 오른쪽으로 이동을 뜻한다.
- done은 현재 에피소드(episode)가 끝났는지 끝나지 않았는지를 나타내는 boolean 값이다. (막대기가 쓰러지거나 카트가 중앙에서 너무 멀리 이동하면 episode를 종료(terminate)한다.

파이토치(PyTorch)를 이용해서 DQN 에이전트 구현하기

- 파이토치(PyTorch)를 이용해서 DQN 에이전트를 구현해봅시다.
- 9강_pytorch를_이용해서_dqn_에이전트_구현하기.ipynb
- https://colab.research.google.com/drive/1ESc36t1KWab_wmBBWUVOG8wIk9l6Qemg?usp=sharing