강화학습으로 슈퍼 마리오 깨기

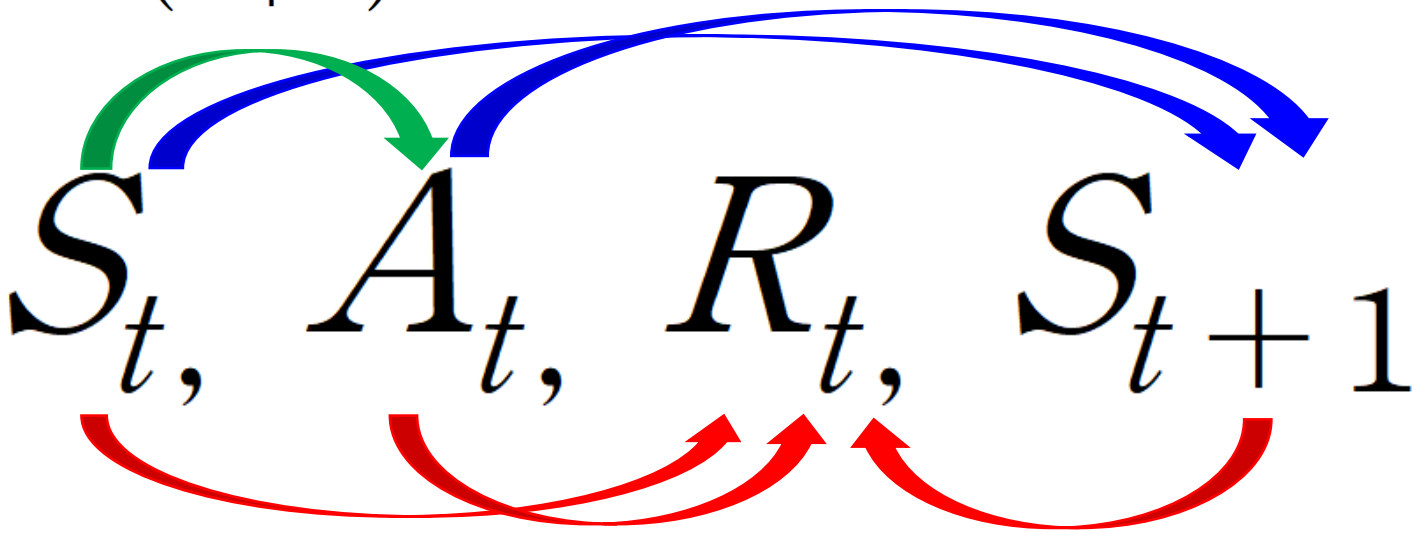
Self Wish
김도훈 박용훈

배경지식(1) – MDP & Bellman Expectation/Optimality Eq.

- MDP
  - State, Action, Reward로 구성됨
  - 각 변수는 상태 전이 확률, 정책, 보상 함수로 도출됨
  - MP에 속하므로, Markov property에 따라 미래 State의 조건부 확률 분포는 과거의 확률 분포와 독립이며, 오직 현재의 확률 분포에만 영향을 받는다는 가정에 따라 정의됨

1. <정책> $\pi(a|s)$

2. <상태 전이 확률> $p(s'|s,a)$



3. <보상함수> $r(a,s,s')$

배경지식(2) - Bellman Expectation/Optimality Equation

- State-value function, Action-value function (Q-function)
  - 상태 가치 함수(State-value function) : 에이전트가 특정 정책을 따를때, 현재 상태에서의 기대 수익
  - 행동 가치 함수(Action-value function) : 에이전트가 특정 정책을 따를때, 현재 상태와 행동에서의 기대 수익
- Bellman Expectation/Optimality Equation
  - 특정(혹은 최적)정책에서 t시점과 t+1시점에서의 상태 가치와 행동 가치 사이의 관계를 나타내는 방정식

0단계

$$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$$
$$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$$

1단계

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$$
$$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, v_\pi(s')$$

2단계

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, v_\pi(s') \right)$$
$$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s')\, q_\pi(s', a')$$

0단계

$$v_*(s_t) = \max_a \mathbb{E}[r_{t+1} + \gamma v_*(s_{t+1})]$$
$$q_*(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a')]$$

1단계

$$v_*(s) = \max_a q_*(s,a)$$
$$q_*(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, v_*(s')$$

2단계

$$v_*(s) = \max_a \left[r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, v_*(s') \right]$$
$$q_*(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s', a')$$

<Bellman expectation eq. / Bellman optimality eq.>

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

<Return>

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$
$$= \mathbb{E}_\pi[G | S_t = s] \text{ (State Value Function)}$$

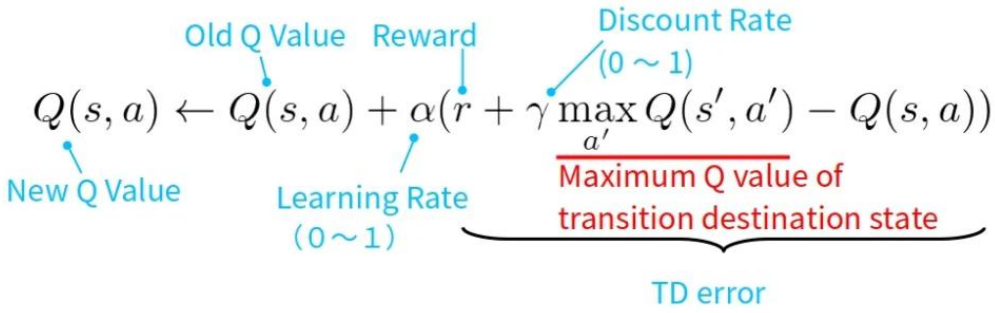
$$q_\pi(s) = \mathbb{E}_\pi[G | S_t = s, A_t = a] \text{ (Action State Value Function)}$$

<State-value / Q-value>

알고리즘(1) : Q-learning & DQN

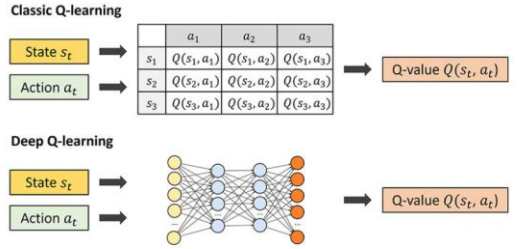
- Q-learning
  - 최적 행동 가치 함수(Q-function)를 안다면, 이를 사용해 최적 정책을 얻을 수 있음
  - 따라서 에이전트는 미래에 Q가 최적 정책의 Q로 수렴할 것이라는 가정 하에 바로 다음 스텝의 Q로 현재의 Q를 업데이트함
  - 부트스트랩의 일종으로, 벨만 최적 연산자가 수축 사상이므로 바나흐 고정점 정리에 따라 최적 정책의 Q로 수렴한다는 것이 증명됨

Ref.) Tuan Anh Le, Proof of Convergence of the Bellman Operator, https://www.tuananhle.co.uk/notes/bellman-convergence.html



<Q-learning>

Ref.) https://www.tcom242242.net/en/entry/rl/qlearning/

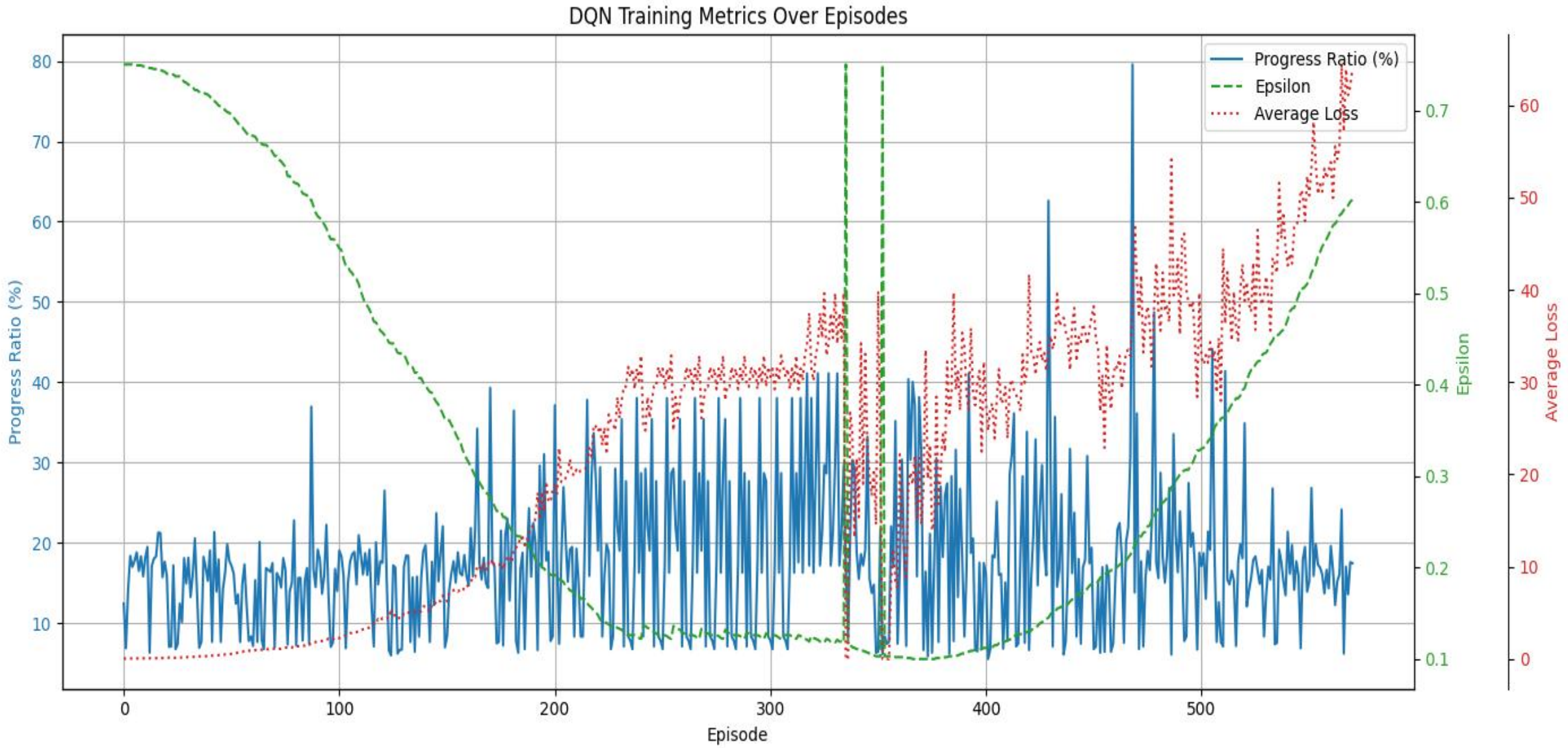


<Q-learning vs. Deep Q-learning>

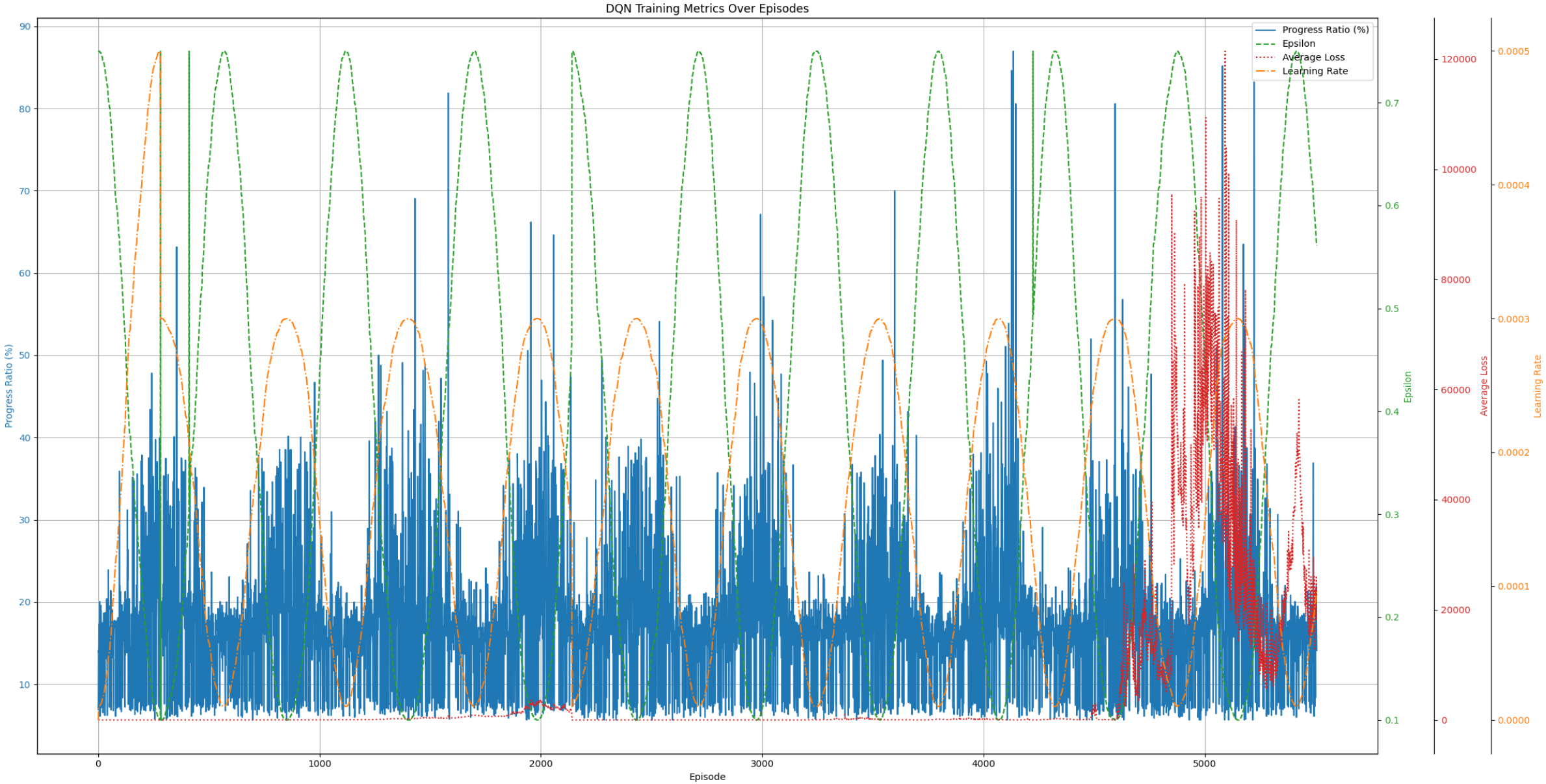
- DQN
  - 기존 Q-learning은 모든 state, action에 대한 모든 Q값의 table을 만들어 Q를 저장할 필요가 있어 효율적이지 못함
  - Q-learning는 목표가 되는 target Q를 정답 레이블로 잡고, 이에 자신의 Q를 회귀시키는 문제로 볼 수 있음
  - 따라서, 현재의 Q값을 신경망으로 구하고, 일정한 주기로 복사된 신경망으로 target Q를 구해 신경망을 회귀시킴
  - 이때, 과거의 MDP(s, a, r, s')를 버퍼에 여러개 저장하고, 이중 일부를 랜덤으로 샘플링해 target Q를 구하는 것에 사용함
  - 또한, 행동 선택 시에는 ε-greedy를 통해 행동(argmax Q)나 탐험(Action중 랜덤 샘플링)을 하여 다양한 MDP를 수집함

Ref.) 사이코 고키, <밑바닥부터 시작하는 딥러닝 4>

Result(1)


DQN Training Metrics Over Episodes
Progress Ratio (%)
Epsilon
Average Loss
Episode
0
100
200
300
400
500
10
20
30
40
50
60
70
80
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50
60

Result(2)

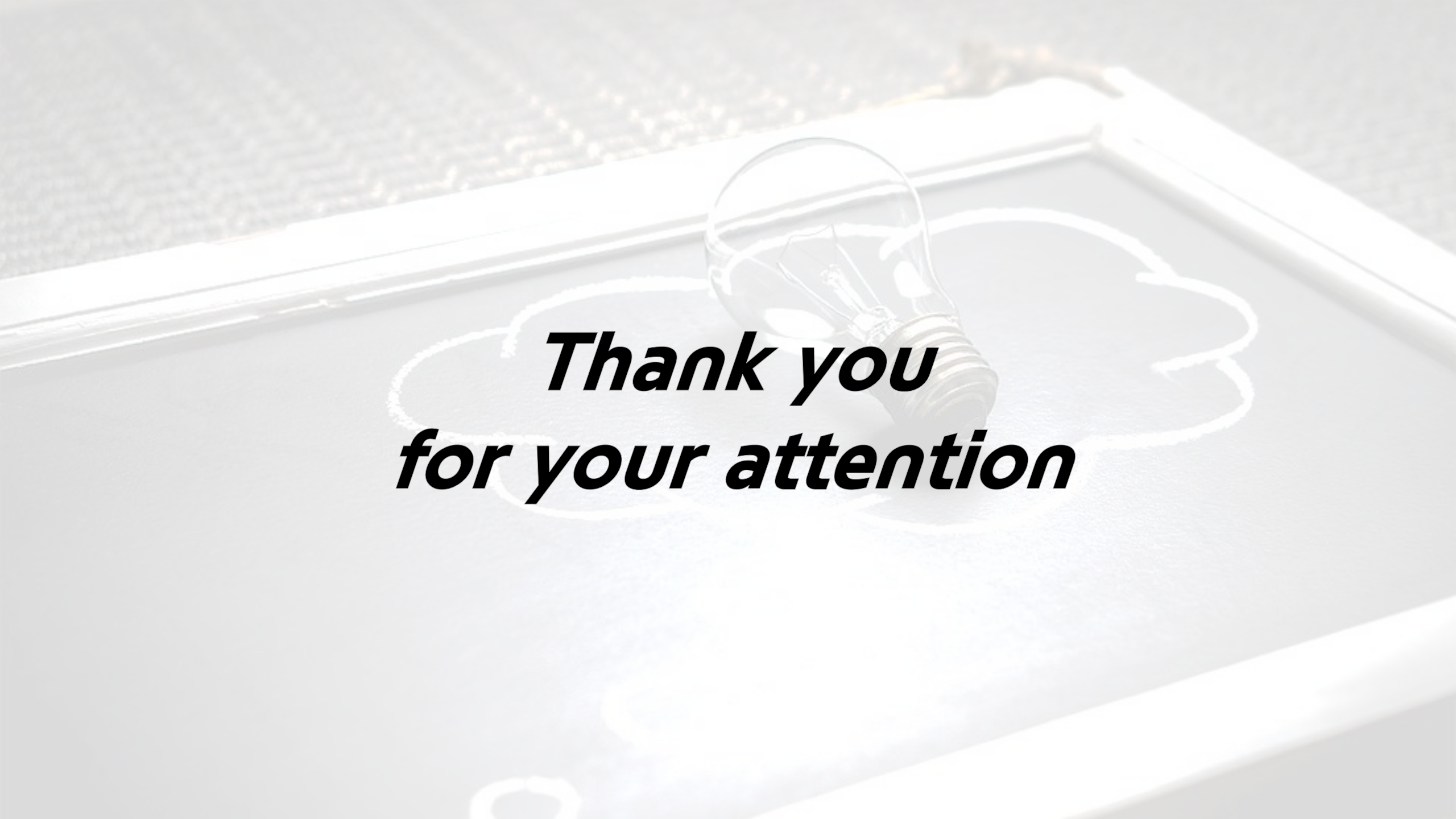

DQN Training Metrics Over Episodes
Progress Ratio (%)
Epsilon
Average Loss
Learning Rate
Episode

Result(3)


```
CUSTOM_MOVEMENT = [
    ['right'],
    ['right', 'B'],
    ['right', 'B', 'A']
]
```

```
JUMP_DURATION = {
    1: 10,
    2: 20,
    3: 20
}
```

Result(3)


Thank you
for your attention