복습)

Q-learning의 핵심

Q에는 1) 내가 있는 상태 state, 2) 내가 하는 행동 action input과 출력 3)quality(reward)가 있다.

frozen lake게임에서 process

Q(s1,left)=0, Q(s1,RIGHT)=1.5 Q(s1,UP)=0, Q(s1,Down)=0.3 이 제공된다면 max value를 선택한다. max의 argument RIGHT를 선택한다

Max Q = $\max_{a'} Q(s,a')$ Q가 가질 수 있는 최대값 a'

$\pi^*(s) = \operatorname{argmax}_a Q(s,a)$ Q가 최대값이 되게끔 하는 argument a -> optimal Π

Q러닝의 가정

S'에서 Q가 있어야 된다.

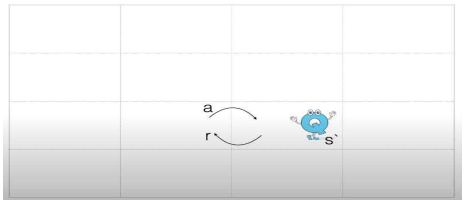
s에서 a를 하고 s'으로 갈거다. a를 하면 r을 받는다

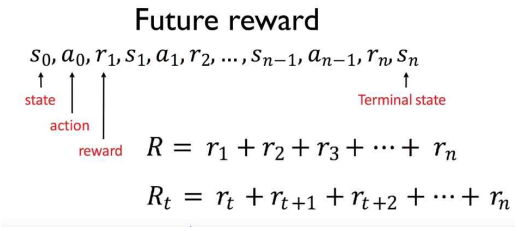
Q(s,a)이라면 Q(s',a')이 존재할거다

Q = r + maxQ(s',a') 이게 되고 이 Q를 학습하는 거다.

현재 Q(s,a)=?

S에서 Q = r + maxQ(s',a')





Reward의 총합은 R = r1+r2+r3....rn

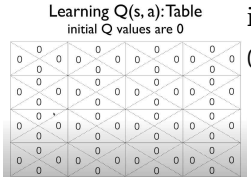
R_t+1 = r_(t+1) + r_(t+2) + r_(t+3) + ... r_n

R_(t) = r_t + R_t+1 가 된다.

R*_t = r_t + maxR(t+1) 가 되고, 결국

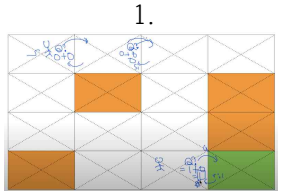
Q(s,a) = r + maxQ(s',a') 가 된다.

frozen lake 문제에서 순서

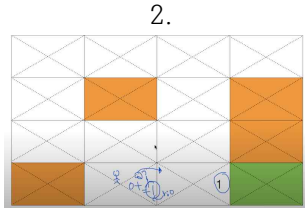


initial Q-table 0으로 초기화

1.

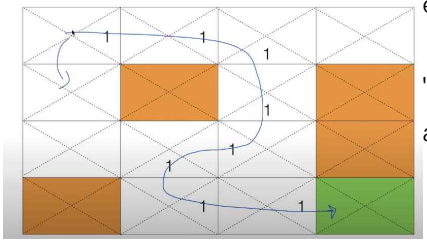


2.



1 -> 2 -> ... n번 진행하면서 각 상태의 Q를 update함

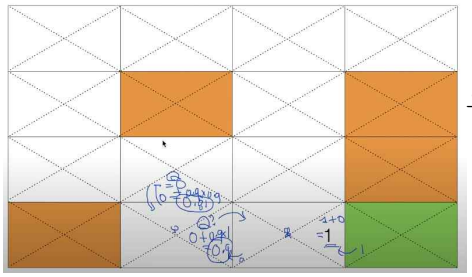
Exploit VS Exploration



exploration을 통해 가본 길 말고 새로운 길을 탐색한다

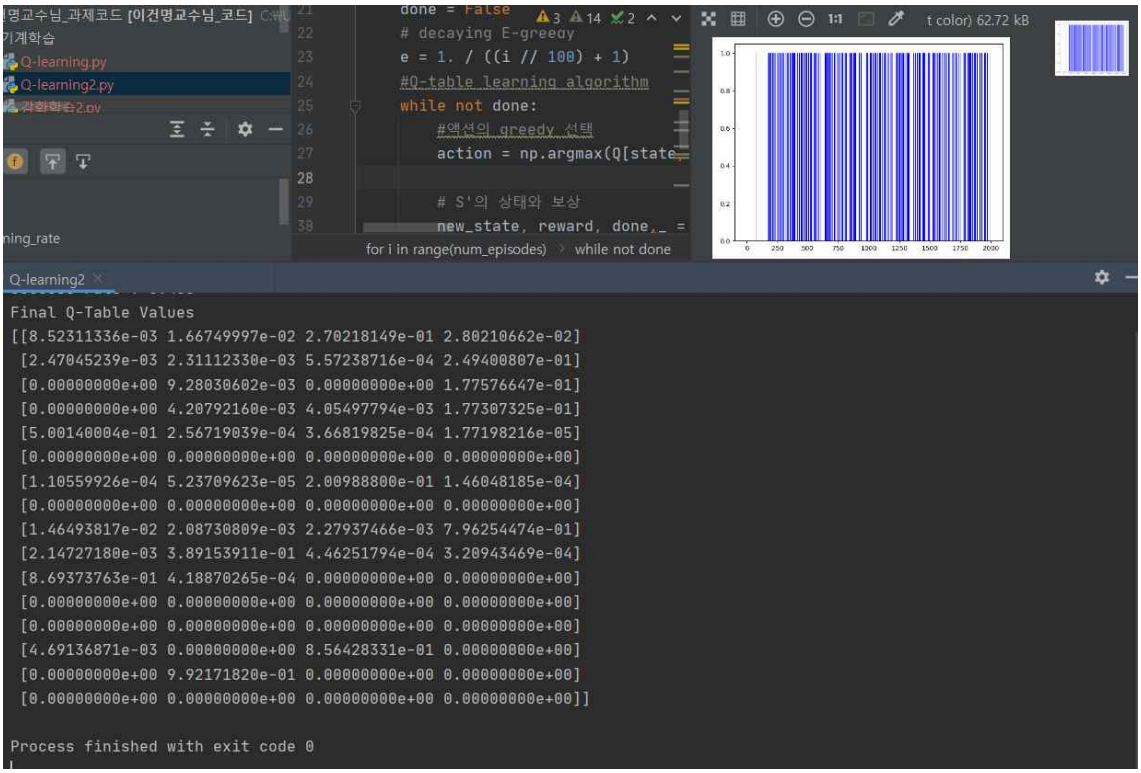
"Select an action a and execute it"
a를 선택할 때 exploit이나 exploration을 선택한다!

Discounted future reward
R_t = r_t + Γr_(t+1) + Γ^2r_(t+2) + Γ^3r_(t+3) + Γ^4r_(t+1) ...



discount factor는 간단하게 미래의 보상을 현재의 값으로 받아들이기 위한 장치이다.

frozen lake 문제에서 e-greedy를 사용하여 action 선택하는 Q-learning
[코드 실행 결과]



Success rate : 0.553

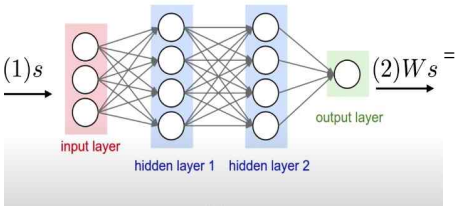
deterministic일땐 r + Γ maxQ(s',a')이 Q이고
확률적(미끄러운 lake) 일땐 (1- Γ)Q(s,a) + Γ(r + Γ maxQ(s',a'))

100*100에서 Q-table을 사용할 수 있을까?
100*100*4(action) = 40000개

실제 80*80pixel + 2color에서는?
2^(80*80)이다
Q-table로는 현실 세계의 문제를 못 푼다

Q-network

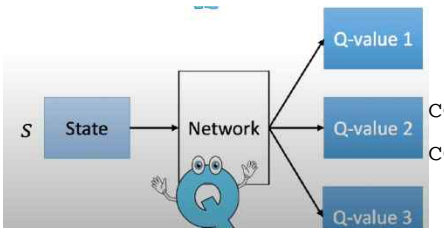
상태 하나에 모든 가능한 액션값의 Q-value을 찾는다



= Q_prediction

$$cost(W) = (Ws - y)^2$$

Q*= $$y = r + \gamma \max Q(s')$$



cost(W) = (Q_pred - Q*)^2
cost를 minimize하도록 학습하는 것이 목표이다

Q^ = Q hat = 예측값
Q^(s,a|Θ) Θ=weight 즉 weight에 따라서 s,a가 달라진다.

-> W_s = Q_prediction = Q^(s,a|Θ) ~ Q*(s,a)

수학적 기호 표기

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$$

Q pred = target

cost(W) = (Q_pred - Q*)^2 이다.
이제 cost를 최소화 하는 Θ를 구하는 것이 문제이다.

deep Q learning Algorithm

Q 네트워크의 w를 초기화 한다

1번째 상태를 가져온다. 그리고 전처리 한다 (ϕ) [단순히 S라고 생각하면 된다]

for

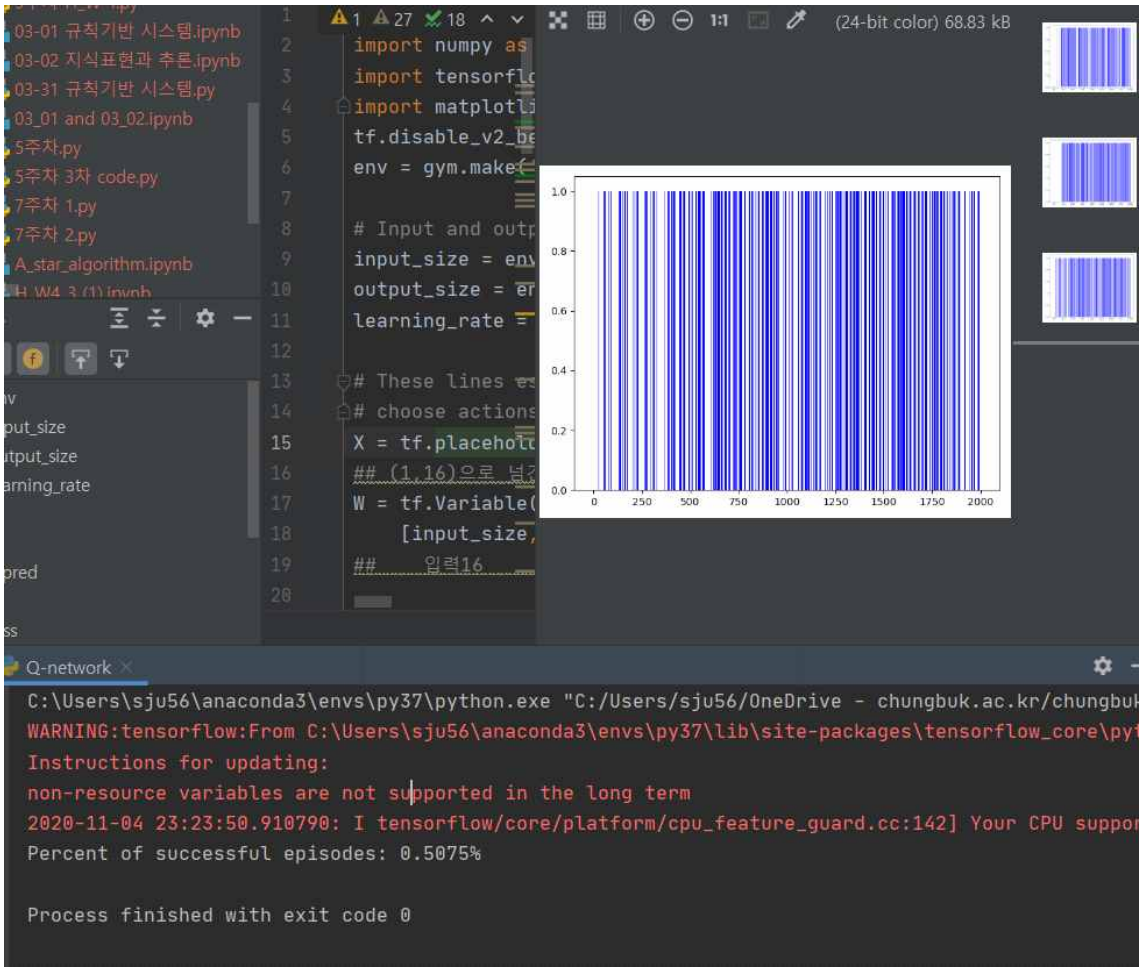
e그리디로 랜덤하게 action을 취하거나 현재 Q에서 max값을 고른다

그리고 a을 한다. 그리고 S_t+1 와 r_t를 받는다

$$y_j = \begin{cases} r_j & \text{마지막 일 때} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{마지막이 아닐 때} \end{cases}$$

그리고 y_i를 구하고 cost값을 최소화 하는 θ를 찾으면 된다 $(y_j - Q(\phi_j, a_j; \theta))^2$

11월 2주차 코드 완성 후 결과



Q-table보다 성능이 안 좋다. 0.5%정도 나온다.

이유 : Q-network에서는 층이 하나였다. ->deep learning은 층이 여러 개로 해야 잘 나온

다. 신호등 인식과 차선 인식 yolo에서는 x, y, 층이 1000개 였다.

2.sample들의 correlations 연관성이 많다. ->minibatch와 같은 상태들을 랜덤하게 선택해서 최적해를 찾아야 한다???

3. Q_pred와 Q_target의 parameter(θ)값을 다르게 하고 1000번, 2000번에 한번씩 target의 parameter를 prediction의 parameter로 update시킨다

->1,2,3, 해결법으로 Q-network를 upgrade시킨 것이 deep mind의 DQN이다.