**소프트웨어 특강2 HW3**

2017312576 이동준

MountainCar-v0 환경

state : agent의 위치(position)와 속도(velocity)를 표현하는 원소로 가지는 배열

action: velocity값을 더 작게 할 것인지, 그대로 둘 것인지, 더 크게 할 것인지 결정(3개)

reward : 각각의 step마다 -1

최종 목적지에 도달하거나 에피소드의 step이 200에 도달하면 해당 에피소드를 종료한다.

1. DQN

\_\_init\_\_: epsilon 초기값을 1로 설정. 학습시킬 model 생성.

\_build\_network : state size(2) -> 512 의 dense layer과 512 -> action size(3) 의 dense layer로 구성된 모델. Optimizer은 Adam optimizer을 사용(learning\_rate: 0.001)하고 loss함수는 MSE를 사용.

predict: Q model network로 해당 state에서 greedy하게 action을 취함.

train\_minibatch: batch\_size는 512로 설정. Replay buffer인 history에서 batch\_size 만큼을 random sampling으로 뽑아내어 minibatch 생성, 그 것으로 state, action, reward, next\_state, done 배열을 생성. Minibatch를 통하여 Q 값을 업데이트(를 하고 업데이트 된 Q를 이용하여 모델을 학습시킴.

update\_epsilon : epsilon값을 0.995배를 취한다.(단, 0.01 밑으로는 내려가게 하지 않는다.)

learn: 매 에피소드마다 environment를 reset.

해당 state에서 epsilon(eps)의 확률로 random한 action. 1-epsilon의 확률로 greedy한 action을 취한다. 해당 state, action과 action 이후 얻은 next\_state, reward, done를 replay buffer(history)에 저장.

replay buffer의 크기가 LEARNING\_STARTS보다 커지면, train\_minibatch함수를 실행하여 학습시킴. 이후 state, step\_count, eps를 다음 step으로 넘어가기 전에 수정.

한 episode가 끝나면 eps값을 이전 episode의 1step을 기준으로 0.99배가 되도록 초기화(0.01밑으로 내려가지 않게)

2. double DQN

\_\_init\_\_: DQN과 동일한 과정 후 evaluation으로 사용할 모델 eval생성(model은 학습용)

\_build\_network : DQN과 동일

predict : DQN과 동일

train\_minibatch: DQN의 과정에서 Q 값을 업데이트 할 때, (로 하고(Q’은 eval 모델) model에 Q를 학습시킴.

update\_epsilon: DQN과 동일

update\_eval: eval을 model과 동일하게 업데이트.

learn: 시작할때 eval을 model과 동일하게 update DQN과 동일한 과정 중 한 episode가 끝나면 update\_eval함수를 eval을 업데이트.

3. multistep DQN

\_\_init\_\_: DQN과 동일 + n\_step을 4로 설정

\_build\_network : DQN과 동일

predict : DQN과 동일

train\_minibatch: DQN과정 중 Q를 (으업데이트 하여 model에 Q를 학습시킨다.

update\_epsilon: DQN과 동일

learning: DQN과 동일한 과정 중 replay buffer(history)에 state, action, reward, next\_state, done 값을 저장 후 뒤의 n\_step들에도 저장.

4. PER

구현 X