소프트웨어특강2 : 강화 학습 개론

Assignment 3: Deep Q-Networks

1. 목표

- Open AI Gym MountainCar-v0 환경에서 DQN을 구현하여 문제 해결 하기
- Double Q-learning, Multistep을 추가 구현하여 성능을 비교해 보기
  (Prioritized Replay Buffer 추가 구현한 경우 보너스 점수 획득)

2. 개발 환경

- Python (version 3.5+)
- Open AI gym (0.15.4)
- **Tensorflow (2.0.0)** – 만약 tensorflow version 1.x 를 사용하려면 tensorflow 2.0을 설치 후 "**import tensorflow.compat.v1 as tf"**로 사용하면 됩니다.

3. 템플릿 파일 및 제출 파일

- 제공되어지는 템플릿 파일 3개
  - dqn.py : DQN 클래스 (보너스 PER 구현도 이 파일에 내에 구현해 주세요) 및 해당 파일을 자유롭게 수정 가능 (제출용). 단 script.py 파일과 호환이 되어야 함.
  - script.py : 위에서 만든 DQN 으로 MountainCar-v0 를 학습시키고 그 결과를 그래프로 출력해주는 프로그램. (제출할 필요 없음)
- 보고서 : 과제 환경(OS, pip version 등), 구현 알고리즘, 결과 그래프 설명을 작성하여 제출.

4. Open AI gym : MountainCar-v0

- A car is on a one-dimensional track, positioned between two "mountains". The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum.
- env = gym.make('MountainCar-v0')

  state_size = env.observation_space.shape[0] **=> 2**

  action_size = env.action_space.n **=> 3**
- State (observation) : Box(2)

  Shape가 (2,)인 continuous space로써 각각 velocity와 position을 나타냄
- Action space = Discrete(3)

  (범위가 0~2인 정수 값, 각각 0 : push left, 1 : no push, 2 : push right을 나타냄)
- Reward : 매 step 마다 항상 -1, goal position에 도착 시 episode 종료 (done)
- 환경이 reset( ) 되면, 최저점의 위치에서 속력이 0인 상태로 초기화합니다
- 총 1500 에피소드를 학습 시켜 매 에피소드 마다 최근 100 에피소드의 step 평균을 기록하여 learn( ) 함수가 결과를 반환합니다
- env.render( )를 사용하면 아래와 같이 실행 화면을 창으로 보여줍니다

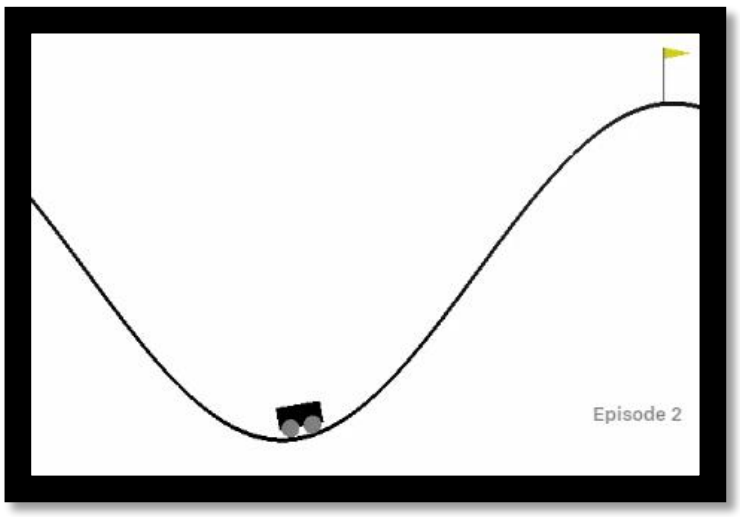

그림 1. MountainCar-v0 실행 화면

5. dqn.py 설명

*** dqn.py 파일의 DQN 클래스를 수정합니다.**

*** DQN class 이름, learn( ) 함수명, 파일명 이외에는 모두 수정/추가가 가능합니다.**

- 클래스 설명

Class DQN:

def __init__(self, env, double_q=False, multistep=False, per=False):

✓ 변수 double_q, multistep, per은 각각 Bool 값으로써, 추가적으로 구현한 Double, Multistep, PER를 사용할지 사용하지 않을 지 결정하는 변수들입니다.

✓ Ex) Multistep을 구현에 성공했을 시, script.py 파일에서 **DQN(Multistep=True)로 DQN 인스턴스를 생성하면 Multistep이 적용된 DQN** 모델이 생성되어 학습을 진행하며, **DQN(Multistep=False)로 생성하면 Multistep이 적용되지 않은 DQN**이 작동해야 합니다. 이를 유의하여 프로그래밍 하시기 바랍니다.

✓ Learn( ) 함수를 호출하면 학습이 시작됩니다. Learn( ) 함수는 script.py 에서 호출하기 때문에 이름을 바꾸면 안됩니다.

✓ action = self.predict(state)

- predict 함수를 호출하여 현재 policy에 따른 action을 반환합니다.
- 템플릿 코드에는 항상 랜덤 액션을 반환하도록 설정되어 있습니다.

✓ next_state, reward, done, info = env.step(action)

- env의 현재 상태에서 action을 수행해 next_state, reward, done, info 정보를 받음

✓ 에피소드의 reward 평균을 기록한 내용은 learn( ) 함수의 반환 값으로 넘겨줍니다.

6. script.py 설명

*** script.py 파일은 수정할 수 없습니다. 또한 제출할 필요가 없습니다.**

- script.py에서는 옵션 (파일 실행 시 입력) 에 맞게 여러 개의 DQN 모델들을 생성하여 학습하는 프로그램입니다. 또한, 학습된 결과 (Episode에 따른 평균

reward 변화)를 그림 3의 그래프로 출력해주는 프로그램 입니다.

- 채점 시, TA는 **해당 script.py를 수정 없이 그대로 실행할 예정입니다.** 따라서 script.py는 제출할 필요가 없습니다.

- **실행 명령어**

```
$ python script.py default
```

```
$ python script.py default double multistep per
```

그림 2. script.py 실행 예시

- Ex) Default DQN, Multistep DQN 두 개를 실행 후 결과를 비교하고 싶은 경우

  $ python script.py **default multistep**

  위처럼 실행시키면 기본 DQN 과 multistep 2개의 DQN 모델을 생성하고 학습을 진행한다. 그리고 2개의 실험 결과만 그래프를 생성해 보여준다.

- 실행의 최종 결과로써 아래와 같은 그래프가 result.png로 저장한다.

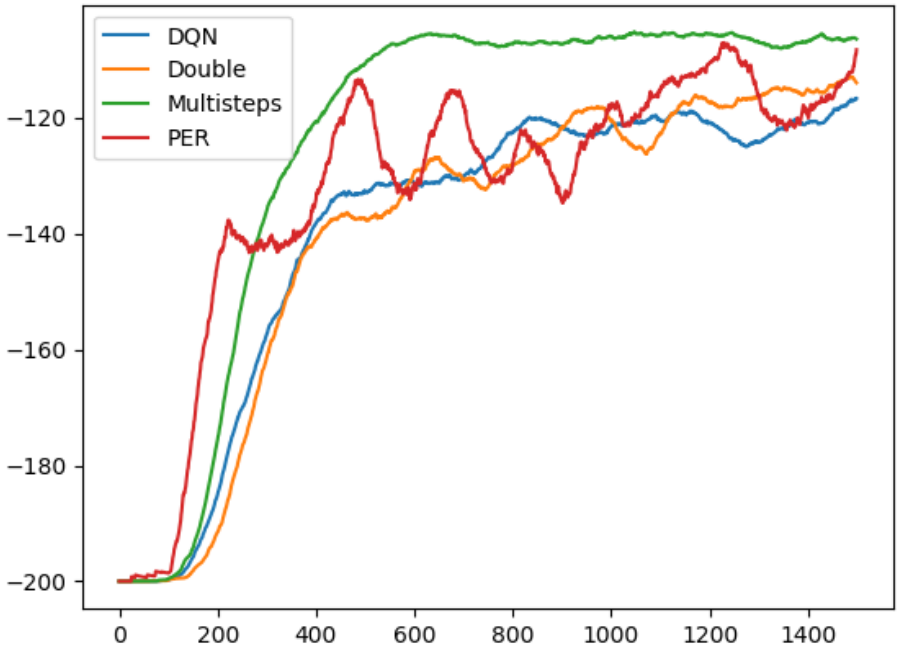

그림 3. Result.png 예시

7. 제출관련

- **제출 마감일 : 12.4 (수) 23:59**
  - **지연 제출은 24시간 마다 -15 점, 72시간 초과는 받지 않습니다.**
  - **표절은 양쪽 모두 0점 처리 되며, 두 번 이상 반복 시에는 F 입니다.**
- **"dqn.py" 파일 명을 그대로 사용하고 보고서 파일과 함께 zip으로 압축하여 iCampus로 제출합니다.**
- **인터넷(github 등)에 존재하는 코드를 그대로(혹은 변수명만 바꾸어) 제출하는 경우 표절도 검사를 통해 불이익이 있을 수 있습니다.**

8. 채점 기준

- Total 100 points
  - 70 points : 기본 DQN이 학습 되는 경우
  - 10 points : Double DQN이 학습 되는 경우
  - 10 points : Multistep DQN이 학습 되는 경우
  - 10 points : 보고서(환경 설명 및 구현 알고리즘 설명 필수)
  - + 5 points : PER DQN이 학습 되는 경우