CartPole 예제도 마찬가지로 Qnet 적용해보자.

random\_uniform 쓰던거, 이렇게 바꿔줬음 – xavior initializing

        initializer = tf.initializers.GlorotUniform()

        self.W = tf.Variable(initializer(

            shape = (input\_size, output\_size)))#weight initializing?

my\_model에서 float 64를 float32로 캐스팅하려 한다는 warning이 뜬다.

학습 시도를 해보는데 딱히 학습이 잘되지는 않는다. 강의에서도 평균 몇십 step정도 reward를 받던데, 성능 자체는 정상적이다. 문제는, 속도가 너무 안나온다는 것이다! 강의에서 학습되는 속도보다 한 20배는 느린 것 같다… 해결법이 없을까? 코드를 이곳 저곳에서 누더기처럼 기워 댔기에 어쩔 수 없는 것일까, 아니면 tensorflow 2.0의 속도가 어쩔 수 없이 이정도 느린걸까.

----속도 문제 관련 구글링 하다가 찾게된 잡설 -------

<https://data-newbie.tistory.com/425> 포스트에 따르면, 요즘 대세는 Pytorch라고 한다. 새로운 논문들이 전부 pytorch 기반으로 나온다고 한다. tf진영 창시자인 Deepmind조차 pytorch로 넘어가고 있는 추세라고. 이게 사실이면…

Tf의 장점은 기업에서 쓰고있고 배포에서 장점이 있다는 것인데, 문제는 TF2는 그런 장점이 하나도 없다는 것이다. 게다가 3월기준 TF2는 성능 메모리 버그 문제들로 고통받고있다는데 어…

timeit을 이용해 걸리는 시간을 측정해보았다. 문제가 되는 부분은 생각보다 많은 시간이 걸릴 것이다. 반복되는 구간마다 time stamp를 numpy.array에 더해준 뒤 차이를 구했다. 이 값들을 각각 .sum()해주면 각 구간이 전체적으로 얼마나 시간을 잡아먹는가 확인할 수 있을 것이다. 15 episode동안 구해본 결과…

time12 : 14.157866400000001 (사이클 시작부터 action을 취해 값을 받아 온 직후까지)

time23 : 14.286717899999978 (값을 받아 온 직후부터 optimize할때까지)

whole time: 29.4581684

충격적이게도, 비슷한 시간을 가진다. 균일하게 문제가 있다는 뜻일까? 구간을 조정해서 다시 측정해보았다.

time12 : 16.058064400000035 (action 받아오기 직전까지)

time23 : 16.43890829999998 (action 받고, optimize하는 시간 포함)

whole time: 33.4426092

더 충격적이게도, action을 받아오는 행위 자체는 별다른 시간이 필요하지 않은 것 같다. 얘가 주된 범인일 줄 알았는데?

한칸 더 올려보자… 그랬더니

time12 : 0.004294699999983997(시작~Qs 구하기 위해 modelQpred.predict 하기 전까지)

time23 : 24.76996160000003(predict 이후)

whole time: 25.8092579

prediction이 문제였다!!!

prediction을 tf.matmul(x,modelQpred.W)로 바꿔주었더니, 예전에 만난 에러 (expected to be a double tensor but is a float tensor)가 다시 나왔다. 맞다, prediction 쓴 이유가 이거였지.

modelQpred(x)를 사용했더니

'tensorflow.python.framework.ops.EagerTensor' object does not support item assignment 에러가. Qs가 eager tensor여서 값을 변경하지 못한다는 의미인 것 같은데…?

<https://github.com/tensorflow/tensorflow/issues/40261> 포스트의 tomerk가 남긴 댓글에 따르면 model.predict는 루프 안에서 쓰면 너무 오버헤드가 큰 top-level api라며, 대체품으로 아래 문장을 사용할 것을 추천하고 있다.

model(x), training=False)

training=False 를 넣어보았지만 여전히 같은 오류

오류를 구글링 해보니 tensor가 not assignable하다는 사실에 많은 사람들이 분개해하며 pytorch를 찬양하고 있었다. 어찌되었건, tf variable로 변경 해준 뒤 값을 바꿔줘야 할 것 같다.

variable로 변경해도 assign이 불가하다! Variable.assign을 써야한다는 것 같은데, 차라리 tf.make\_ndarray 를 사용해보자.

여전히 안된다. 절망적이다… eager tensor는 tensor\_shape 속성이 없다며 에러를 밷는다.

.numpy()를 사용하면 된다고 한다!

된다!!! 속도가 무지막지하게 빨라졌다!

어찌되었건, 왜 학습이 잘 안되는가?

network 변수가 4개밖에 없다, 너무 shallow함. +

샘플간 correlation, 타겟의 불안정(non-stationary)이 원인.

DQN으로 들어가보자.

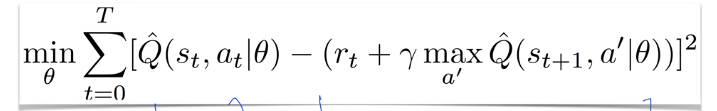
<https://dryjelly.tistory.com/140?category=785786> 참고할만한 tistory post

Nature 표지를 장식한 Deepmind 팀의 DQN 논문.

Correlation between sample의 의미는 무엇인가?

action 1회를 취했을 때 state는 극적으로 변화하지 않는다. 즉, 학습에 필요한 sample(state) 간에유사성이 너무 크다. 샘플이 넓게 퍼져 있으면 학습하기 쉽지만, 비슷한 위치에 몰려 있는 샘플로는 제대로 된 모델이 학습되는 것을 기대하기가 힘들다.

non-stationary targets



같은 네트워크를 사용하기 때문에, Q를 최적화 시키면 target 역시 변경되어버린다. 저번주 정리에 생각했던 문제점.

해결법? 더 깊게, capture and replay, 네트워크 분리.

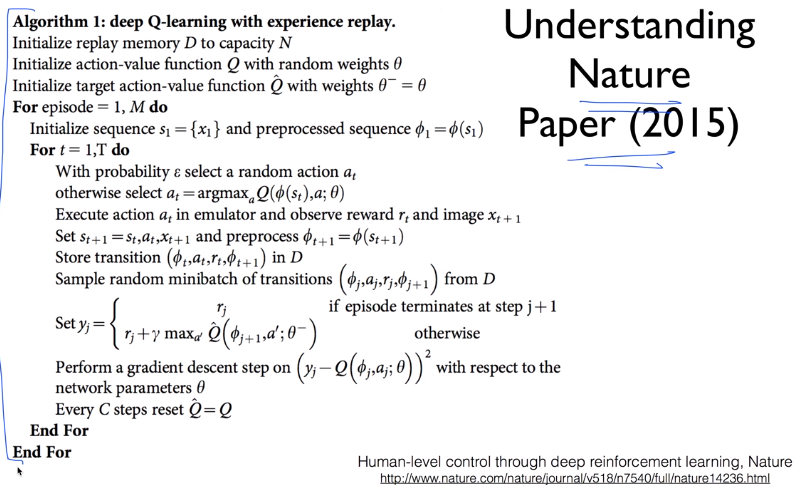
Go deep: convolution network, full connect 등등…

capture and replay : 루프를 도는 도중 동시에 학습시키지 말고, 버퍼에 저장해 놓은 뒤 학습시킨다. 이 때 저장된 transition들 중 랜덤하게 뽑아 minibatch로 만든다.



단 두줄로!

Separate target : 네트워크를 하나 더 만든다. 기본 Qnet 하나와, Target 도출용 network 하나. Target network를 고정시킨 뒤 Qnet을 먼저 학습시키고,일정 C 스텝 마다 이 둘을 동기화 시켜준다.



1 리플레이 저장할 버퍼 D 생성

2 action-value function생성

3 target action-value function 생성

4

5 state 초기화 및 전처리

6

7~8 E greedy 기반 action 선택

8

9 action 취하고 r, 이미지 획득

10 state바꿔주고 전처리

11 저장

12 랜덤 minibatch 뽑은 뒤

13

14 target 을 구하고

15

16 target과 비교해 학습-일단시타만

17

18 시타 바 에 동기하ㅗ

19

20

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

DQN까지 tensorflow2로 스스로 만들어 보기? 차원이 다른 난이도가 되지 않을까 싶다. 그래도, 시도해 볼까?