<https://github.com/hunkim/ReinforcementZeroToAll> TF1버전 강의 코드자료모음

06\_q\_net\_frozenlake.py TF1 버전을 TF2버전으로 바꾸려면 무엇이 필요한가?

TF2에서 주어진 loss function을 쓰는 것만 해왔는데? 이걸 직접 만드는 방법이 필요하다?.

place holder, variable을 사용하는 방법을 배워야한다?

함수형 API를 사용하면 간단하게 해결???

<https://tensorflow.blog/2019/03/06/tensorflow-2-0-keras-api-overview/> TF2 고급사용법

위 링크 내용, 지금까지 배웠던 tf2사용법 등등을 모두 참고해서 드디어 tf2버전 qnet frozenlake를 만들었다. 그런데 문제는 학습이 너무 오래걸린다… 어림잡아 30분은 걸려야 될 것 같은 속도이다. 학습 할 때 마다 일일히 출력해서 그런걸까? model.fit에서 출력빈도를 낮추는 방법을 찾아봐야겠다.

학습이 느릴 뿐더러, 학습이 전혀 안된다. 뭐가 문제일까? Qnet은 학습이 빨리 되는데… 성공률은 바닥을 친다. 초기 weight가 너무 세게 지정 되어있는 것 같기도 하다.

속도가 느린 것은 dense를 써서 그런건가 싶어 matmul을 사용해보기로 했다.

초기 weigt을 강의버전처럼 줄이고 matmul을 나름 적용해보았는데, 속도는 그대로인것같다. Qnet의 loss는 상당히 작다. 근데 이게 문제가 아닌 듯 싶다… 너무 오래걸린다.

표준출력을 하느라 시간이 오래걸리는걸지도 몰라서 fit에 verbose = 0 를 넣어보기로 했다.

조금 빨리 걸리는 것 같다. 하지만 성공률 0퍼센트를 찍었다. 홀리… 랜덤으로 가도 0%는 아닐텐데 뭐가 문제일까.

input shape을 바꿔줬더니 warning이 사라졌다. 이제 기대해볼만 한가?

전혀…. 성공률 0.002%. 대체 왜이러지?

s,one\_hot(s), Qs등등을 프린트 해보아도 별 문제가 없어 보인다. 정확히 곱하고 싶은 모양 대로 곱해지고 결과가 나오고 있다. 그런데 학습이 안된다는 뜻은 SGD가 문제이거나, loss function mse가 문제이거나 인 것 같다. 이걸 model 상속해서 직접 만드는 방식으로 해결해야 하는걸까.

loss function을 나름 만들어서 넣어줘 봤다. 근데 학습이 안된다. W가 꿈쩍을 안한다. 왜일까.

가설 : fit을 반복시켜놓으면 fit 스스로가 학습 할때마다 w를 리셋하는게 아닐까?

검색해본 결과 그렇지 않다고 한다. 그렇다면 gradient decent 자체가 작동하지 않는다는 이야기이다.

weight이 변화하는지, 변화하지 않는지 보기 위해 copy.deepcopy 메소드를 사용한다. 변수는 변수간 대입을 하면 얕은 복사가 이루어지고, mutable한 변수인 경우엔 같은 주소를 가진 이름만 다른 변수가 되지만 immutable한 변수의 경우는 값을 변화시킬 시 다른 주소를 가진 변수가 된다고 한다. 텐서가 어떤 변수인지 모르니, 새로운 객체를 만들어주는 deepcopy를 그냥 적용하자.

np.array\_equal, equiv로 W와Wtemp(deep copy한 W)를 비교하려 하는데, 복사 직후 비교해도 어째선지 계속 다르다고 나온다. 프린트해서 비교해보면 같은 값에 같은 shape인데 왜일까? type이 달라서인가? type()으로 비교해보니 당연히 같다.

.numpy()를 사용해 텐서를 numpy array로 만들어 비교해야겠다. 이랬더니 비교가 잘되는 것 같다.

<https://www.tensorflow.org/tutorials/customization/custom_training?hl=ko#%EB%8B%A4%EC%9D%8C_%EB%8B%A8%EA%B3%84> 사용자 정의 model, loss, train 제작에 관한 글

나름대로 training, loss, model을 만들었다. (예전 모델은 주석처리하고)

TypeError: \_\_call\_\_() got an unexpected keyword argument 'training'

듣도 보도 못한게 튀어나왔다. predict메소드를 계속 사용하려고 tf.keras.model 클래스를 상속했는데…

class MyModel(tf.keras.Model):

    def \_\_init\_\_(self):

        super().\_\_init\_\_(name = 'my\_model')

        self.W = tf.Variable(tf.random.uniform(

                                [input\_size, output\_size], 0, 1))#weight

    def \_\_call\_\_(self,x):

        return tf.matmul(x,self.W)

뭐가 문제지? layer역할을 하는 게 \_\_init\_\_에 있어야 하나?

상속에서는 \_\_call\_\_이 아니라 call을 써야되는 것 같다.(TF2 고급 사용법 참조) 이랬더니 중간에 model(input)으로 해 놓은 것이 있어 문제가 되었다. predict로 바꿔줬다.

그랬더니 이젠 47번줄에서 오류가 나온다.

AttributeError: 'numpy.ndarray' object has no attribute '\_in\_multi\_worker\_mode

이건…. 뭐가 문제인지 모르겠다. 구글링해도 답이 없다…

tf.keras.model을 상속받지 않고 하는 방식으로 해볼까? 상속을 지우고, \_\_call\_\_로 다시 바꾸고, predict를 전부 바꿔주고

어딜 잘못고쳤는지 matmul에 float을 넣지 말라고 한다.

cannot compute MatMul as input #1(zero-based) was expected to be a double tensor but is a float tensor

확인해보니 one\_hot() 결과물은 float형이 맞다. 근데 전에는 안그랬으면서 왜 지금은…?

predict를 지워서 이런 문제가 발생했으니, predict가 type을 맞춰주는 기능을 해줬다고 밖에 설명할 수 없는 것 같다.

찬찬히 보면 볼수록, gradient\_tape를 사용한 train부분이 너무 블랙박스다. 이해도 잘 안하고 무작정 쓴 것 같은 느낌이다.

잠시만… train에 왜 MyModel을 쓰고 있었지? modelQperd를 써야되는거 아닌가? 바꿔줬다.

이번엔 다른 에러. 그것도 가장 긴 에러…

극한의 고통… train을 포기하고, optimizer 중 하나를 가져오는게 낫지 않을까 생각된다.

<https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Optimizer> 참고한 optimizer 사용법

48%! 드디어! 성공했다!