LAB QUESTION C

Team ASIC

Q1. Modify and2.py to perform 2-input OR function.

#1.

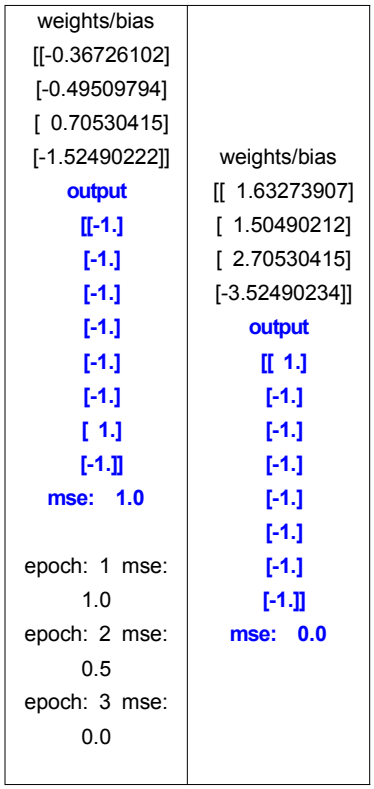
```
weights/bias
[[-0.04962436]
[ 0.22194259]
[ 0.30185649]]
output
[[ 1.]
[ 1.]
[ 1.]
[ 1.]]
mse:  1.0

epoch:  1  mse:  1.0
epoch:  2  mse:  2.0
epoch:  3  mse:  0.0

weights/bias
[[ 1.95037568]
[ 2.22194266]
[ 2.30185652]]
output
[[ 1.]
[ 1.]
[ 1.]
[-1.]]
mse:  0.0
```

<Result>

A. 2-input OR function 프로그램의 output은 train in과 w의 행렬 곱을 step function을 이용하여 표현함으로써 도출 가능하다. train in은 진리표의 X, Y input 값을 나타내며, w는 3*1 random array를 나타낸다.
위의 결과를 분석하면, 첫 번째 test()에서 4행 1열의 [1.] 성분이 error였으므로, 1.0의 mse(mean square error)가 나타났고, 3번째 epoch에서 0.0으로 개선된 것을 확인 할 수 있다. 그 결과, 두 번째 test()에서 output은 [[1.], [1.], [1.], [-1.]] 으로 정상 출력되었다.

Q2. Modify and2.py to perform 3-input AND function.

#1.



<Result>

A. 3-input AND function 프로그램은 w를 4*1로 수정하고, train in과 train out을 각각 3개의 입력에 대한 출력으로 변경함으로써 연산가능하다. 3개의 입력을 가질 때 AND function의 진리표는 다음과 같다.

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

진리표와 대조하여 위의 결과를 분석하면, 첫 번째 test()에서 1행 1열의 [-1.] 성분과 7행 1열의 [1.] 성분이 error 였으므로, 1.0의 mse(mean square error)가 나타났고, 3번째 epoch에서 0.0으로 개선된 것을 확인 할 수 있다. 그 결과, 두 번째 test()에서 output은 [[1.], [-1.], [-1.], [-1.], [-1.], [-1.], [-1.], [-1.]] 으로 정상 출력되었다.
단, 여기서 주의 할 점은 3-input AND function 프로그램에서 mse는 에러 하나 당 0.5라는 점이다. 이 문제에 대해 교수님께 질문할 필요가 있다.

Q3. Modify the code to use gradient descent.

A. 해당 프로그램에 대한 이해도가 낮아 어느 부분에 대해 gradient descent를 적용해야 하는지 파악하지 못하였습니다. 이 문제에 대해 tensorflow programming을 충분히 복습 및 학습하여 추후에 다시 한번 해결할 수 있도록 시도해보겠습니다.

Q5. Modify and2.py to perform 2-input XOR function.

#1.

weight 1 [[-0.20532468 0.15216459] [1.30696988 0.66990048]] bias 1 [0. 0.] weight 2 [[-0.22179776] [0.88957912]] bias 2 [0.] **output** **[[0.40016615]** **[-0.21856499]** **[0.21856499]** **[-0.40016615]]** **mse: 1.10395** epoch: 1 mse: 1.10395 epoch: 2 mse: 1.08078 epoch: 3 mse: 1.0621 epoch: 4 mse: 1.0473 epoch: 5 mse: 1.03569 epoch: 6 mse: 1.0266 epoch: 7 mse: 1.01945 epoch: 8 mse: 1.01376 epoch: 9 mse: 1.00916 epoch: 10 mse: 1.00536 . . .	epoch: 1344 mse: 0.0102335 epoch: 1345 mse: 0.0102066 epoch: 1346 mse: 0.0101798 epoch: 1347 mse: 0.0101532 epoch: 1348 mse: 0.0101266 epoch: 1349 mse: 0.0101003 epoch: 1350 mse: 0.010074 epoch: 1351 mse: 0.010048 epoch: 1352 mse: 0.010022 epoch: 1353 mse: 0.00999616 weight 1 [[1.09290206 1.55034852] [1.14917862 2.13372183]] bias 1 [-0.6894089 2.06288743] weight 2 [[-2.02866507] [2.04696679]] bias 2 [-1.54916549] **output** **[[-0.87567157]** **[0.91847551]** **[0.92371732]** **[-0.89064312]]** **mse: 0.00997044**

<Result>

A. 2-input XOR function은 Gradient Descent로 구성되며, 이를 이용하여 각 epoch마다 learning rate만큼 mse를 반복하여 보정하는 작업을 거친다. err값이 target보다 커지거나 epoch가 Max epoch에 도달하는 경우 mse 보정을 중지하며, 이 때의 결과는 XOR function의 진리표와 같이 출력된다. 위의 <Result>를 참고하면 첫 번째 test()에서, 전반적인 output matrix의 성분에서 mse가 발생하여 그 값이 1.10395로 출력됐으나, 0.01을 target으로 하여 mse를 보정한 결과 그 값이 0.00997044까지 하락했으며, 이를 이용하여 두 번째 test()를 실행하였을 때의 output은 XOR의 진리표와 유사한 [[-0.8756~], [0.9184~], [0.9237~], [-0.8906]]으로 출력되었다. Output의 값이 정확히 [[-1], [1], [1], [-1]]이 아닌 이유는 mse가 0.0에 도달하지 못하였기 때문으로 유추된다.