**IMDB 데이터셋을 이용한 이진 분류**

컴퓨터공학과 고은비

[실습문제 정의]

IMDB(Internet Movie Database)는 인터넷 영화 데이터베이스로, 총 5만 개의 리뷰가 있다. 5만개의 리뷰 중 각각 50%는 부정, 50%는 긍정 리뷰로 구성되어 있기 때문에 IMDB는 이진 분류의 예제로 활용하기에 적합하다.

실습에서는 데이터 5만 개를 반으로 나누어 절반은 training, 절반은 test에 사용할 것이다. 이 또한 각각의 2만 5천개 데이터 중 절반은 부정, 절반은 긍정 리뷰로 구성된다. 데이터는 전처리되어있어 각 리뷰(단어 시퀀스)가 dictionary의 숫자 시퀀스로 변환되어 있다.

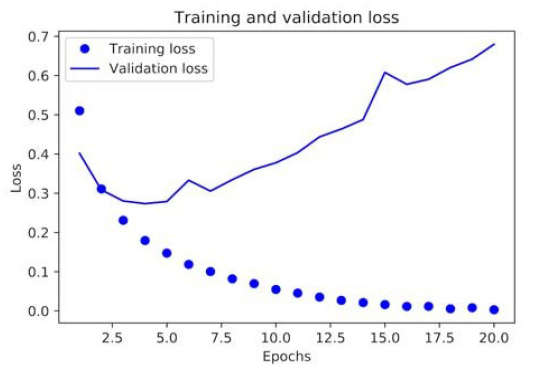
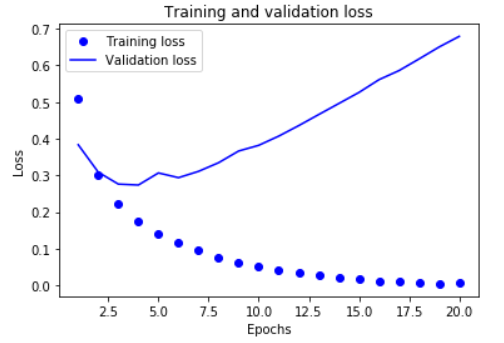
교재에서는 num\_words를 1만 개 사용하였고, 원-핫 인코딩 방식을 이용해 숫자 리스트를 텐서로 바꾸었다. 은닉 층은 relu 활성화 함수를 사용하고, 마지막 층은 sigmoid 활성화 함수를 이용하여 신경망 모델을 구성하였다. 손실 함수는 binary\_crossentropy를, 옵티마이저는 rmsprop를 사용하였다.

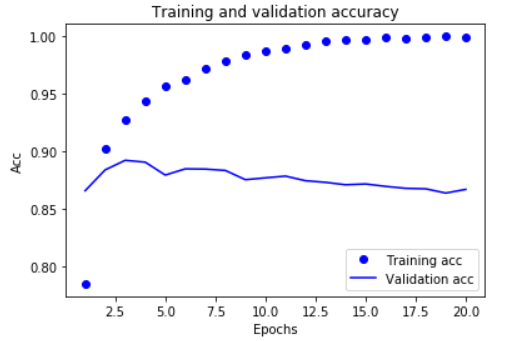
이러한 방식들을 적용한 결과 overfitting 문제가 발생하였다. overfitting이 발생하는 지점을 찾아 그 전까지만 훈련할 수도 있지만, epochs가 너무 적게 수행되지 않도록 만드는 것이 더욱 적절한 학습이자 개선 방안으로 보인다.

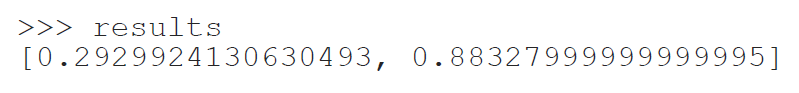
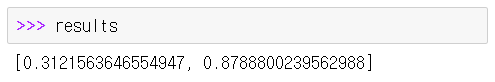
모델이 훈련하기 위해서는 loss function과 optimizer가 필요한데, 이 실습은 이진 분류 문제이고 모델이 확률을 출력하기 때문에 binary\_crossentropy 손실 함수를 사용하는 것이 적합하다. binary\_crossentropy 함수는 확률 분포 간의 거리를 측정한다. 즉, 정답인 타깃 분포와 예측 분포 사이의 거리를 측정한다.

[실습 및 문제 분석]

<교재> <실습-IMDB1>

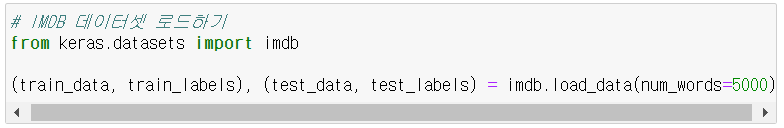
 

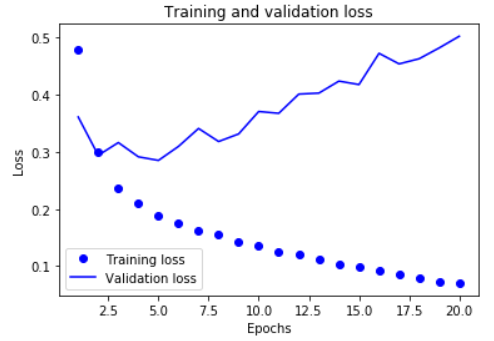
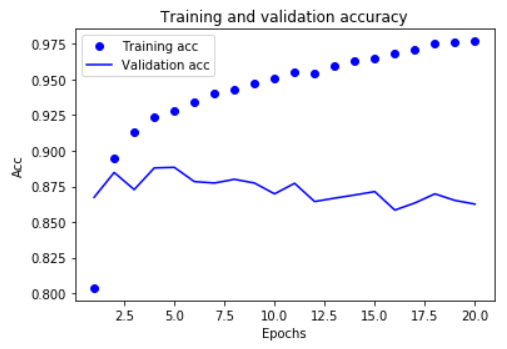
 

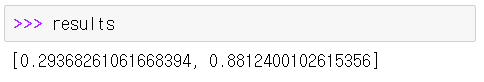
위의 그래프를 보면 알 수 있듯이, 신경망 특성상 그래프가 매 실행마다 동일하게 나오지 않는다. 따라서 그래프의 작은 굴곡들에 집중하기보다는 전체적인 그래프의 형태(상승, 하강)만 파악하면 된다.

교재에서는 overfitting이 epoch6 지점에서 발생하였지만, 신경망의 무작위한 초기화 때문에 위 실습에서는 overfitting이 epoch5에서 발생하였다. 정확도 또한 교재보다 0.5% 낮게 나왔다. 따라서 신경망의 특성상 같은 함수를 사용하더라도 1~2% 정도의 오차는 발생할 수 있다는 것을 알 수 있다.

[실습2 – IMDB2]



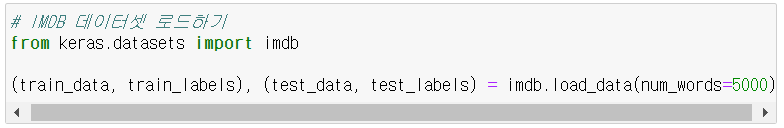
 

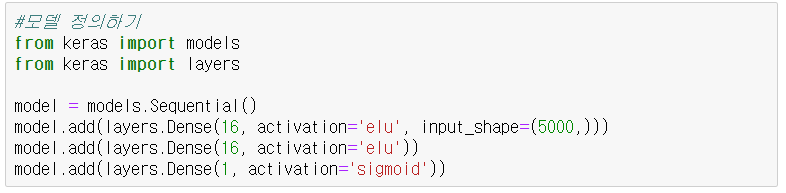


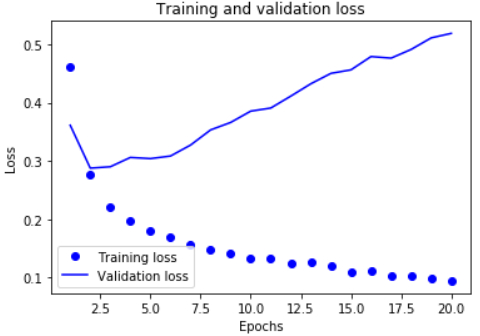
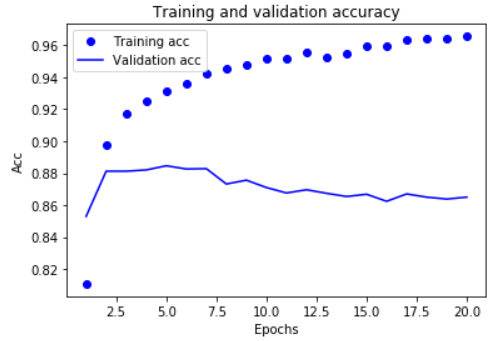
실습2에서는 단어만 5000개로 줄였고, 그 결과 정확도의 차이는 없었다. 이를 통해 단어의 수가 적다고 해서 정확도가 높아지는 것은 아님을 알 수 있다.

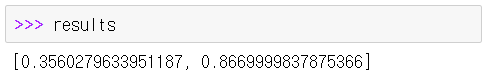
적은 input layer는 training에서는 효과적일 수 있지만, 오히려 overfitting이 발생할 위험이 크며 이는 test에서 더 낮은 정확도를 보인다. 실제로 overfitting이 epoch3 지점에서 발생하였으며, 이는 모델을 훈련하기에 너무 적은 횟수이다.

[실습3 – IMDB3]

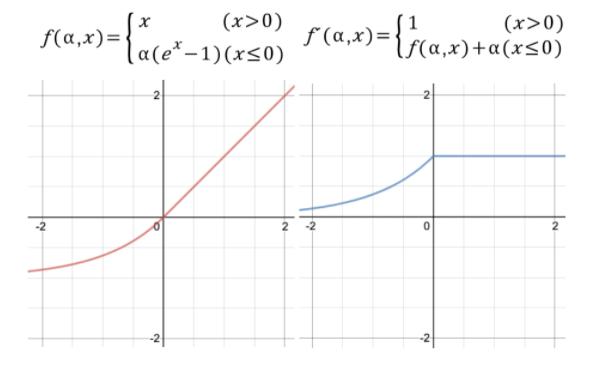




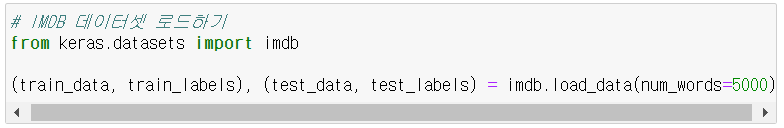


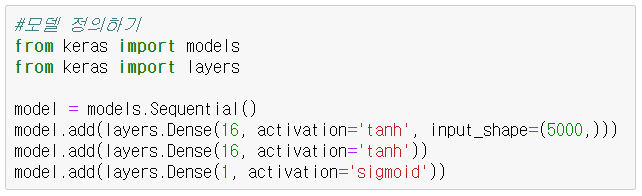
실습3에서는 단어를 5000개로 하고, 활성화함수를 relu에서 elu로 변경하였다. 그 결과 훈련과 검증 정확도가 relu함수에 비해 약간 낮게 나왔고, 정확도가 감소하였다. 하지만 1~2% 정도의 차이는 수행 시 달라질 수 있는 오차값이기 때문에 결과를 분석하여 비교하기에는 부족하다.

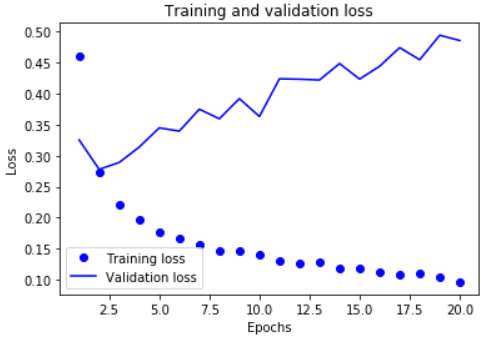
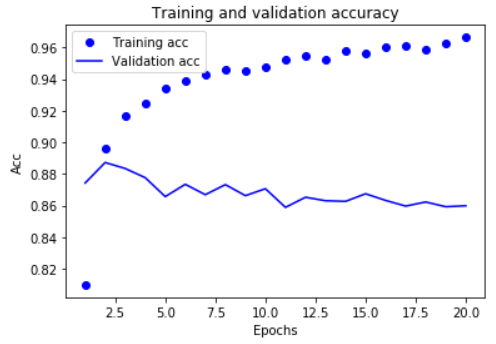


elu함수는 relu함수의 변종으로, relu와는 음수 처리 방식이 다르다. elu함수는 음수를 완전히 제거하지 않고 보통 최대 1까지 출력한다. 지수함수를 이용하며 별도의 알파 값을 파라미터로 받는데 일반적으로 1로 설정된다. 이 경우 미분 함수가 끊어지지 않고 이어져있는 형태를 보인다.

[실습4 – IMDB4]

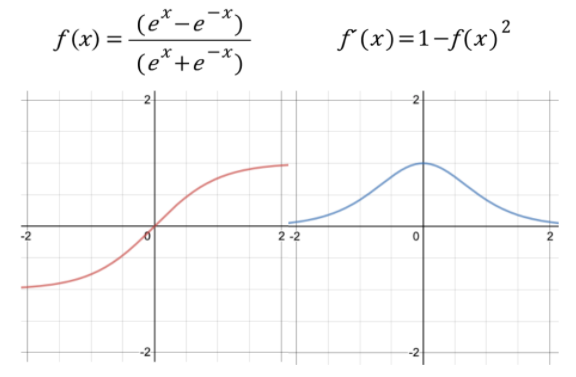




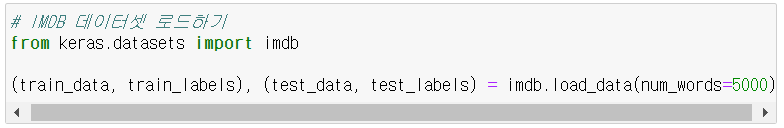


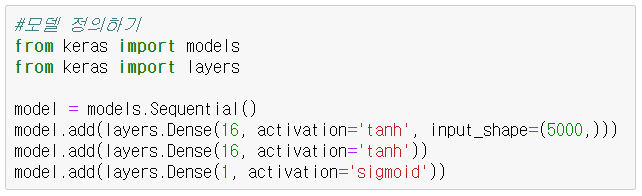
실습4에서는 단어를 5000개로 하고, 활성화함수를 tanh로 변경하였다. 정확도의 차이는 지금까지의 실습에서 크게 차이 나는 경우가 없었고, tanh 신경망 모델과 elu 신경망 모델의 형태가 유사함을 알 수 있다.

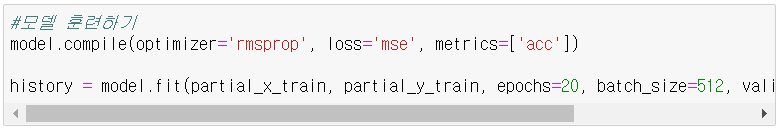


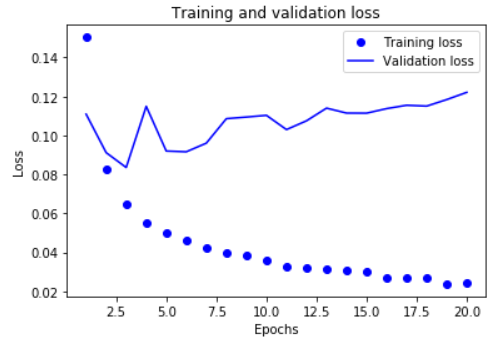
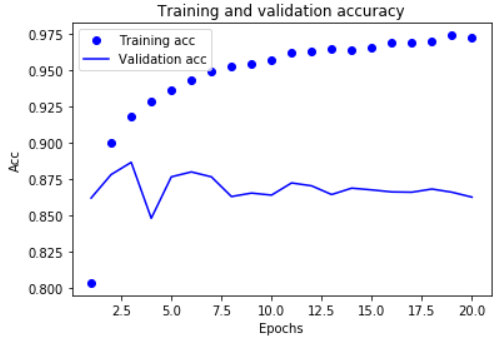
tanh(hyperbolic tangent)함수는 sigmoid 함수와 아주 비슷하지만, 함숫값의 범위가 -1에서 1이다. tanh 함수는 함수의 중심값을 0으로 옮겨 sigmoid의 최적화 과정이 느려지는 문제를 해결한다. 하지만 tanh함수 역시 sigmoid와 마찬가지로 기울기가 사라지는 문제(Vanishing gradient)가 존재한다. 데이터의 평균이 0.5가 아닌 0이라는 유일한 차이밖에 없지만 대부분의 경우에서 sigmoid보다 tanh가 성능이 더 좋다.

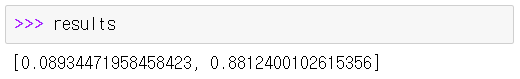
[실습5 – IMDB5]





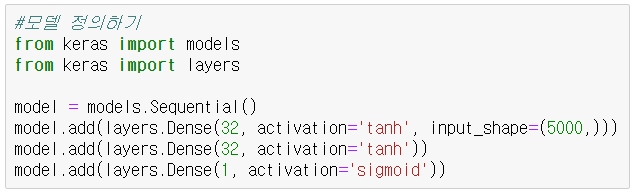


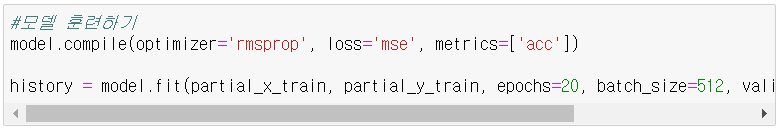
 

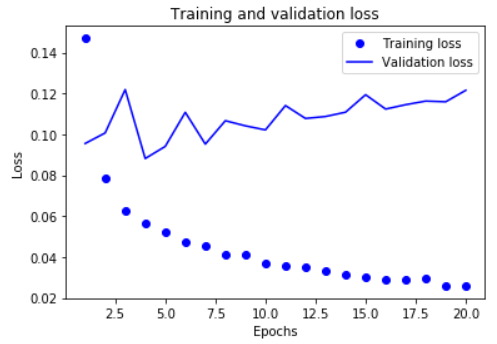
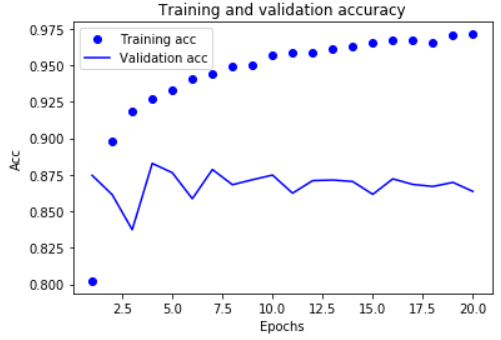


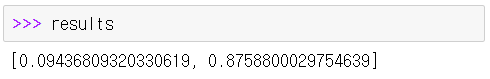
실습5부터는 활성화함수를 tanh, 손실함수를 mse로 해보겠다. 그 결과 훈련과 검증 손실의 차이가 상당히 줄어들었다. 정확도 또한 높게 나왔다. 하지만 mse함수는 이 실습에서 binary\_crossentropy함수와는 달리 훈련된 모델로 새로운 데이터에 대해 예측할 수 없다는 치명적인 단점이 있다.

[실습6 – IMDB6]



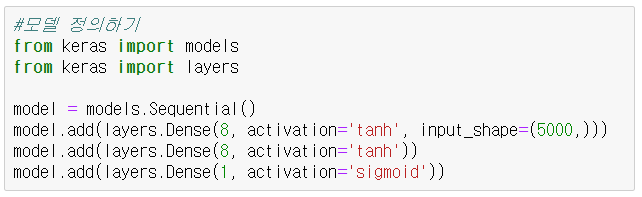


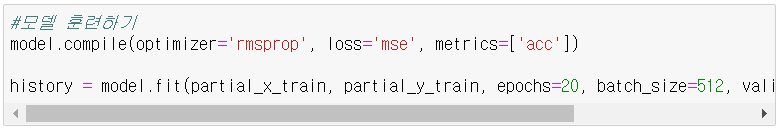
 

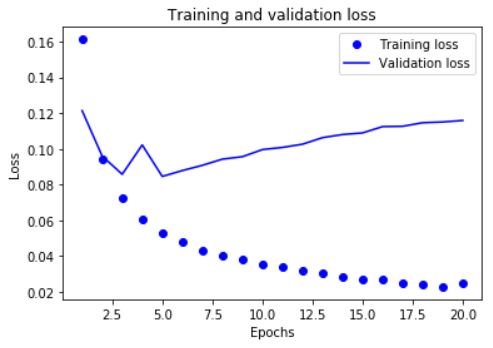
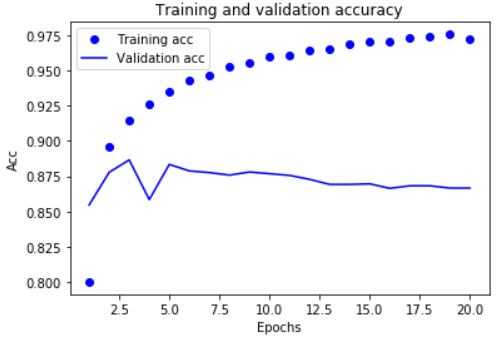


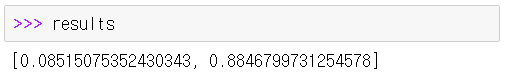
실습6에서는 실습5의 조건에서 layer의 node수만 32로 변경해보겠다. 그 결과 위의 수행과 결과가 특별하게 다른 것이 없었다. 따라서 layer의 node 수가 증가한다고 해서 결과가 크게 달라지지는 않는다는 것을 알 수 있다. 하지만 기본적으로 노드의 개수가 적었기 때문에(16개) 개수를 2배로 늘렸다고 해서 학습에 크게 차이가 없었을 가능성이 크다.

[실습7 – IMDB7]



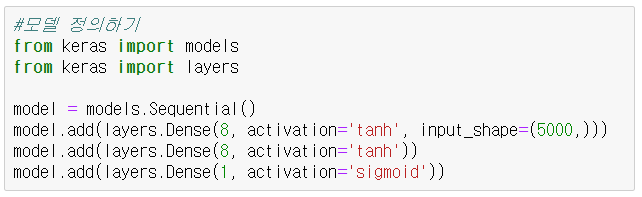


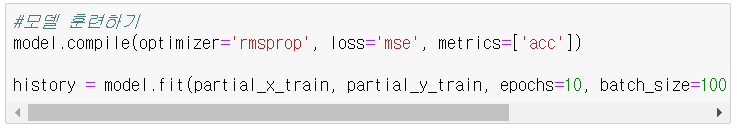
 

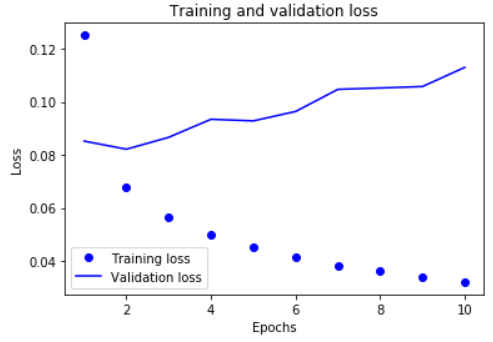
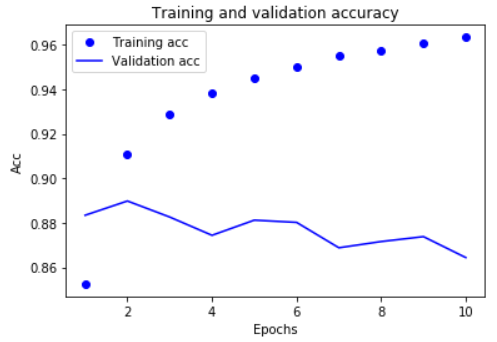


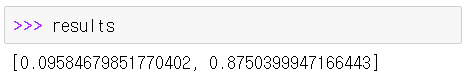
실습7에서는 실습6의 조건에서 layer node수를 8로 변경해보겠다. 정확도가 높은 편이지만 기존 결과와 크게 다르지 않으며, 좋은 학습이라고 하기에는 노드 개수가 너무 적다.

[실습8 – IMDB8]





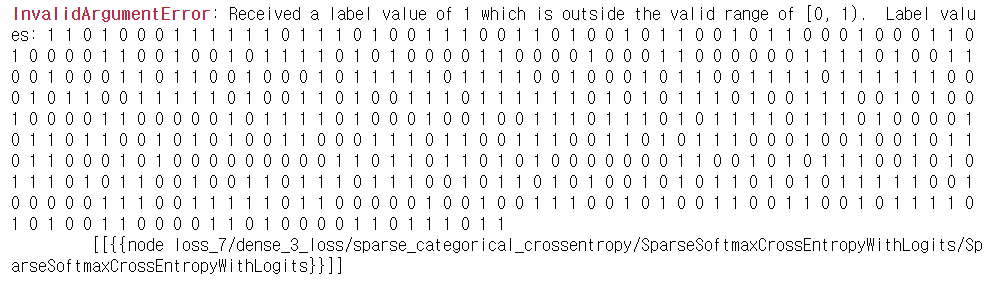
 



실습8에서는 7의 조건에서 epoch 횟수 20->10, batch size 512->100으로 변경하겠다. 훈련을 반복할수록 손실이 줄어드는 것이 학습이기 때문에 epoch 수행을 절반으로 줄였을 시 훈련과 검증 손실은 기존 실습보다 크게 나오는 것이 당연하다. 하지만 그만큼 overfitting을 줄일 수 있다는 장점도 있다.

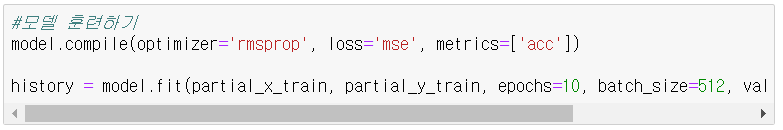
[실습9 – IMDB9]

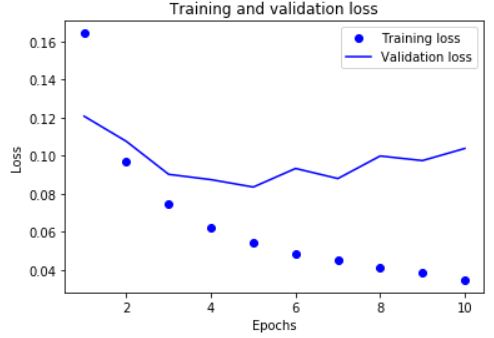
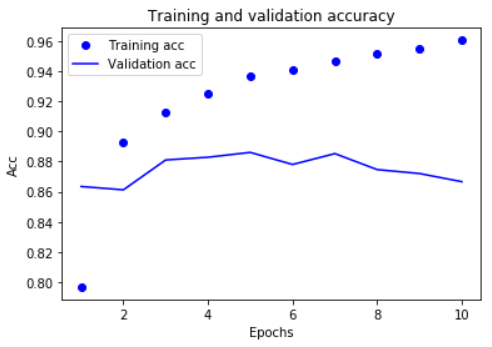


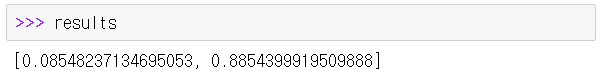


실습9에서는 실습7의 조건에서 손실함수만 sparse\_categorical\_crossentropy로 변경해보겠다. 하지만 sparse\_categorical\_crossentropy 함수는 [0, 1) 범위의 값만 받을 수 있기 때문에 0과 1로 데이터가 처리되어 있는 이 실습에서는 사용할 수 없다.

[실습10 – IMDB10]

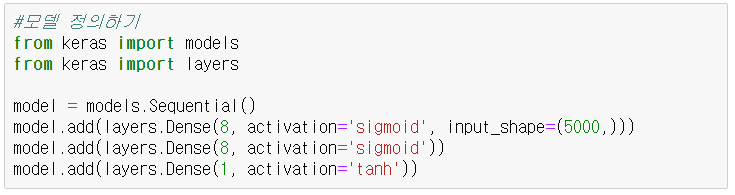


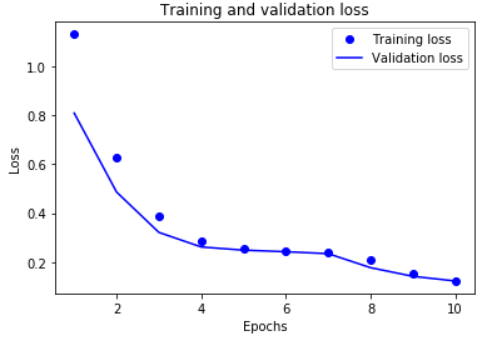
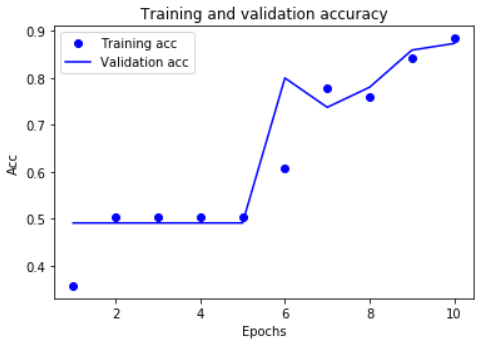
 

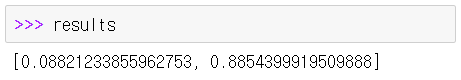


실습10에서는 실습7의 조건에서 epochs 횟수만 20->10으로 변경하였다. 결과는 실습8과 비슷했다. 이를 통해 실습8에서 batch\_size의 변경은 이 실습에서 크게 의미가 없음을 알 수 있다.

[실습11 – IMDB11]





실습11에서는 실습10의 조건에서 활성화함수를 input layer-sigmoid, hidden layer-sigmoid, output layer-tanh 함수로 변경하였다. 결과는 별다른 차이가 없었지만, overfitting이 발생하지 않았다는 점에서 기존 실습과 가장 큰 차이를 보였다. 이는 sigmoid함수의 특징 때문에 발생한 결과로 추측된다. 이 방법은 이진 분류 문제에 효과적이라고는 판단할 수 없지만, overfitting을 막기 위한 방법으로는 적절한 것으로 보인다.

[결론]

위 실습에서 단어 수 변경, 활성화함수의 변경, 손실함수의 변경, layer node 수와 batch size의 변경, epoch 횟수 변경 등 다양한 조건 변경을 수행하였다. 데이터를 실제로 분석하기 위해서는 이진 분류에 최적화된 binary\_crossentropy 손실함수를 사용하는 것이 적합하다. 하지만 다양한 함수를 적용하며 가장 최적의 학습을 찾아나가기 위해서는 위처럼 다양한 활성화함수와 손실함수를 적용하는 것이 좋다고 본다.

교재에서의 학습은 정확도가 88%정도밖에 미치지 못하기 때문에 더 나은 학습방법을 찾아볼 필요가 있다. 따라서 다양한 조건 변경을 통해 여러 결과를 조합해서 위의 실습보다 더 많은 경우의 수를 수행해야 한다.

overfitting문제를 해결하는 것이 우선인 상황이라면, 그에 맞는 함수 등의 조건을 변경하고, 정확도를 높여야 하는 상황이라면 또 다른 경우의 수를 찾아봐야 한다.

[참고자료]

『케라스 창시자에게 배우는 딥러닝』, 길벗

「Activation Function」, https://woolulu.tistory.com/84

「[Keras Study] 3장. 신경망 시작하기 (2)」, https://subinium.github.io/Keras-3-2/

「갈아먹는 딥러닝 기초 [1] Activation Function(활성화 함수) 종류」, https://yeomko.tistory.com/39

「[딥러닝] 활성화 함수 (Activation Function)」, https://cheris8.github.io/deep%20learning/DL-Activation-Function/

「딥러닝에서 사용하는 활성화 함수」, https://reniew.github.io/12/

「케라스와 텐서플로 허브를 사용한 영화 리뷰 텍스트 분류하기」, https://www.tensorflow.org/tutorials/keras/text\_classification\_with\_hub?hl=ko

「활성화 함수(activation function)종류 및 정리」, https://ganghee-lee.tistory.com/32