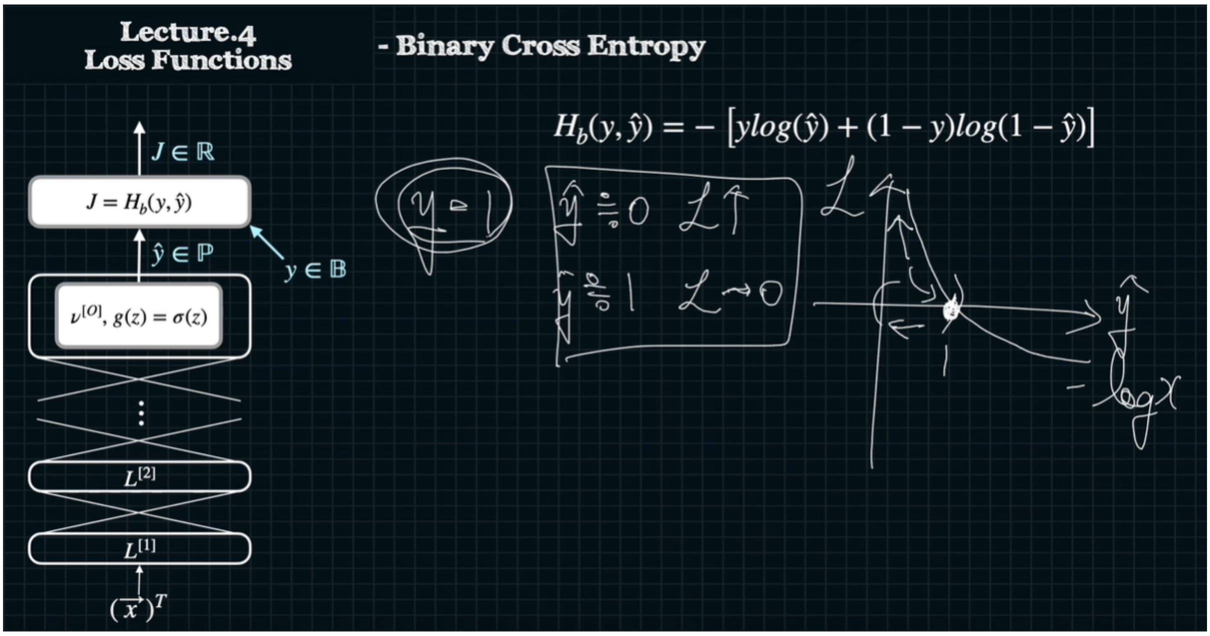
Binary Cross Entropy

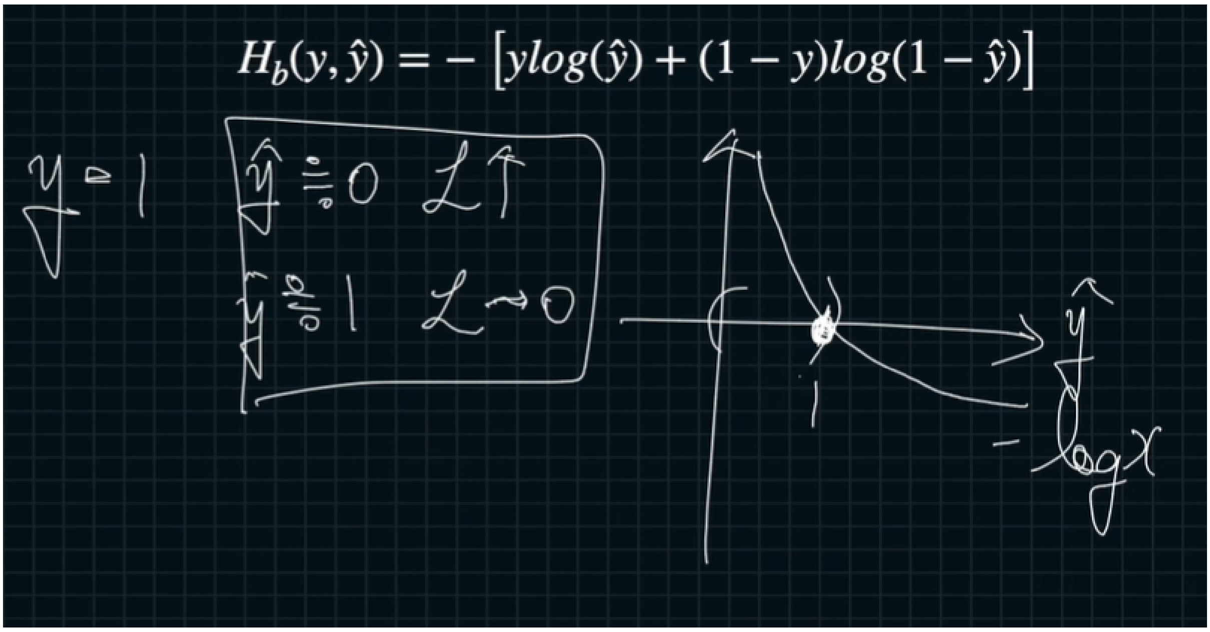


L: loss값은 예측값과 실제값이 비슷하다면 Loss는 0에 비슷해지고

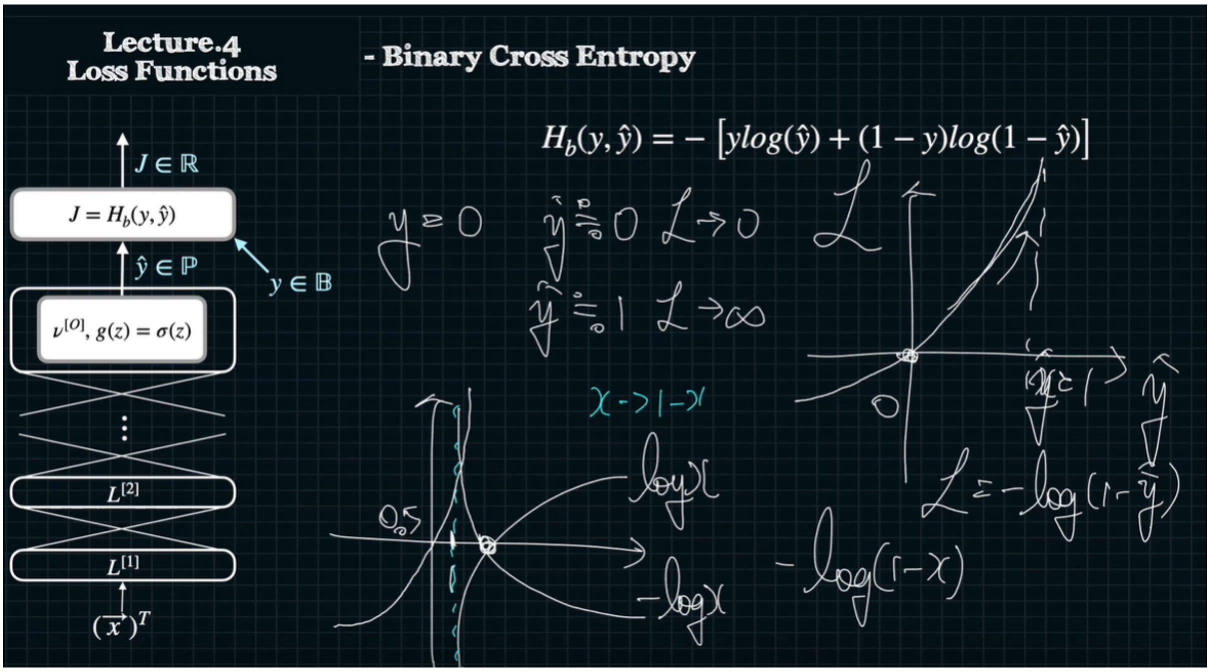
loss값이 예측값과 실제값이 비슷하지 않다면 loss는 무한대로 발산한다.

Ex)

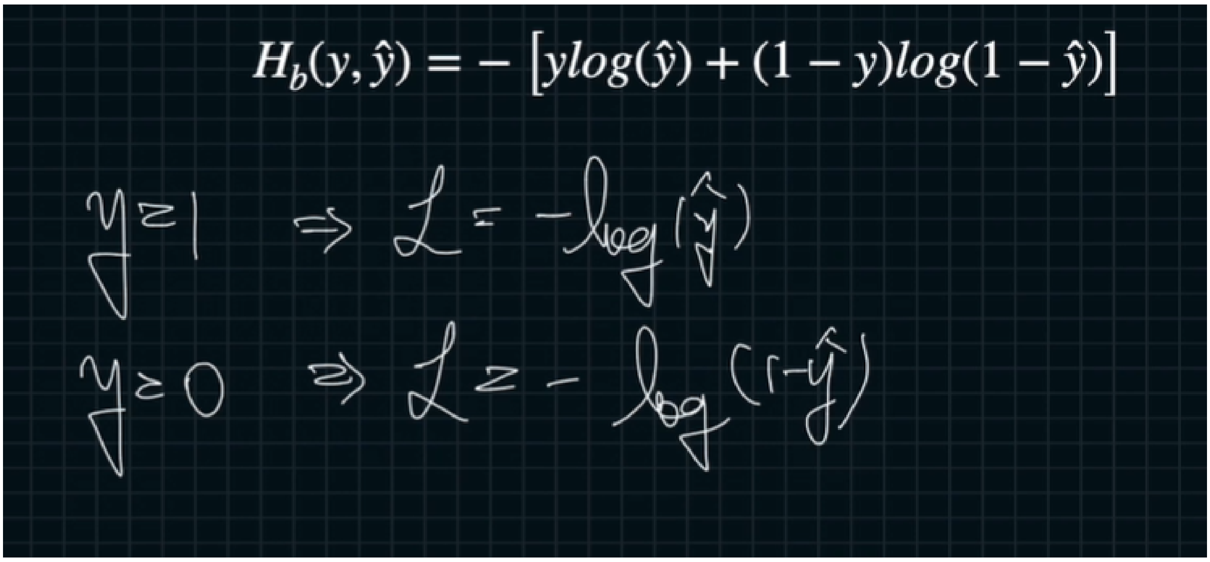
y = 1이라고 가정하고 모델 예측값이 0이면은 Loss값은 무한대로 발산하고 반대로 예측값이 1이면은 Loss값은 0으로 수렴한다.



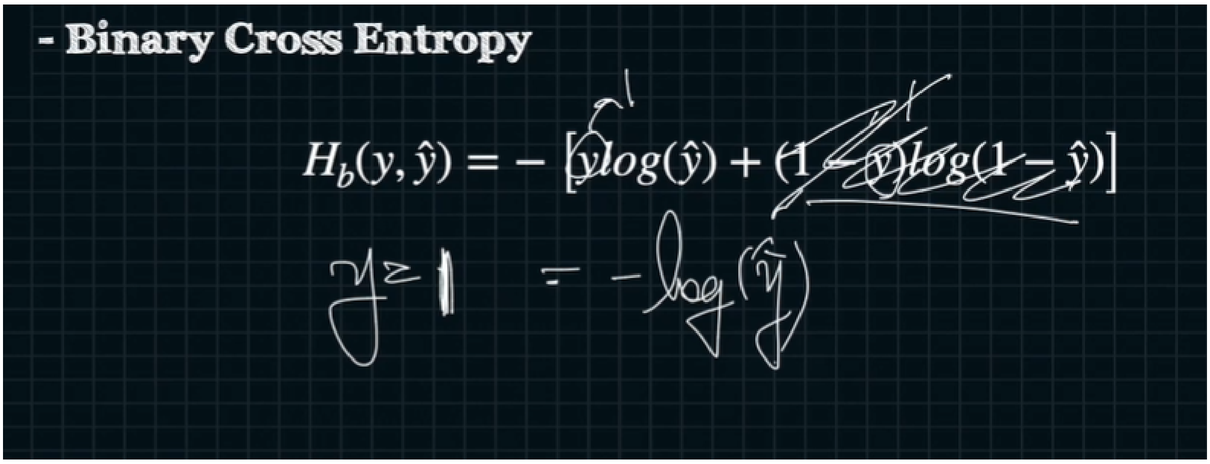
- \- log 그래프는 위 처럼 생겼다. 우리의 예측값은 0 ~ 1 사이에 존재하는데 예측값이 1에 가까워질 수록 0에 가까워지고 반대로 예측값이 0에 가까워지면 무한대로 발산한다.

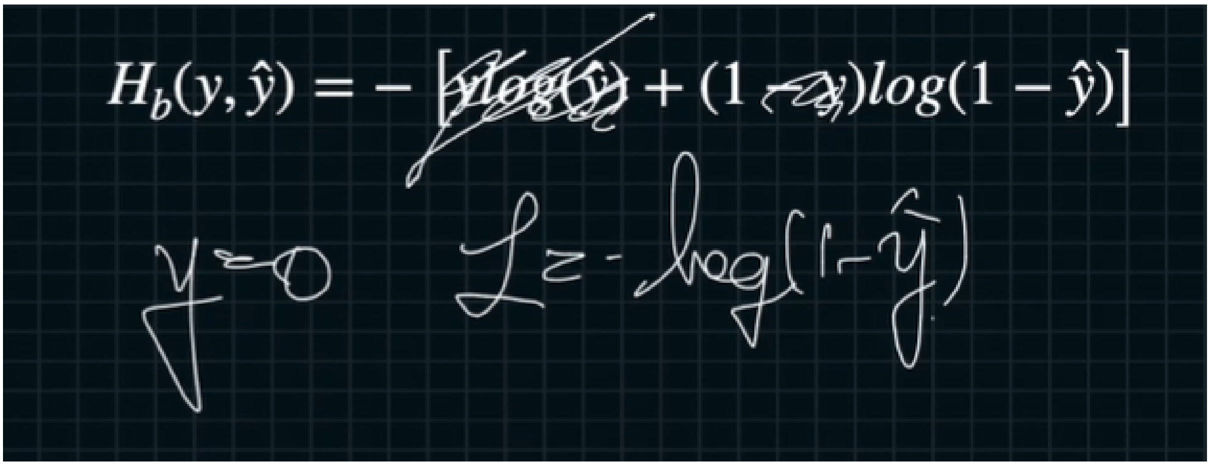


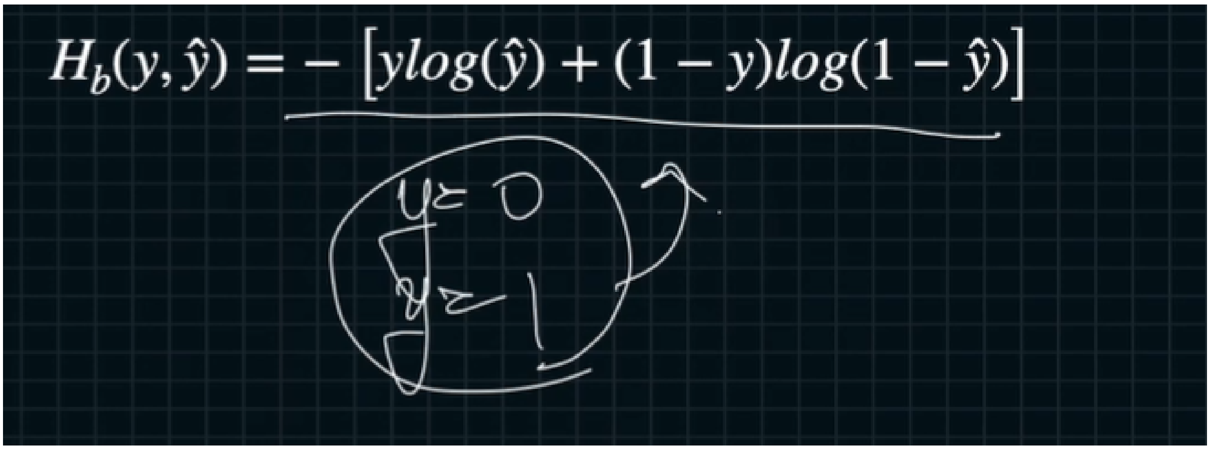
- 실제값이 0일때는 예측값이 0으로 예측하면 loss는 0으로 수렴을하고 반대로 예측값이 1이면은 loss는 1로 발산하게 된다. 그래서 그래프를 그려보면 -logx 그래프의 대칭인 -log(1-x)그래프를 그려지는걸 볼 수 있다.



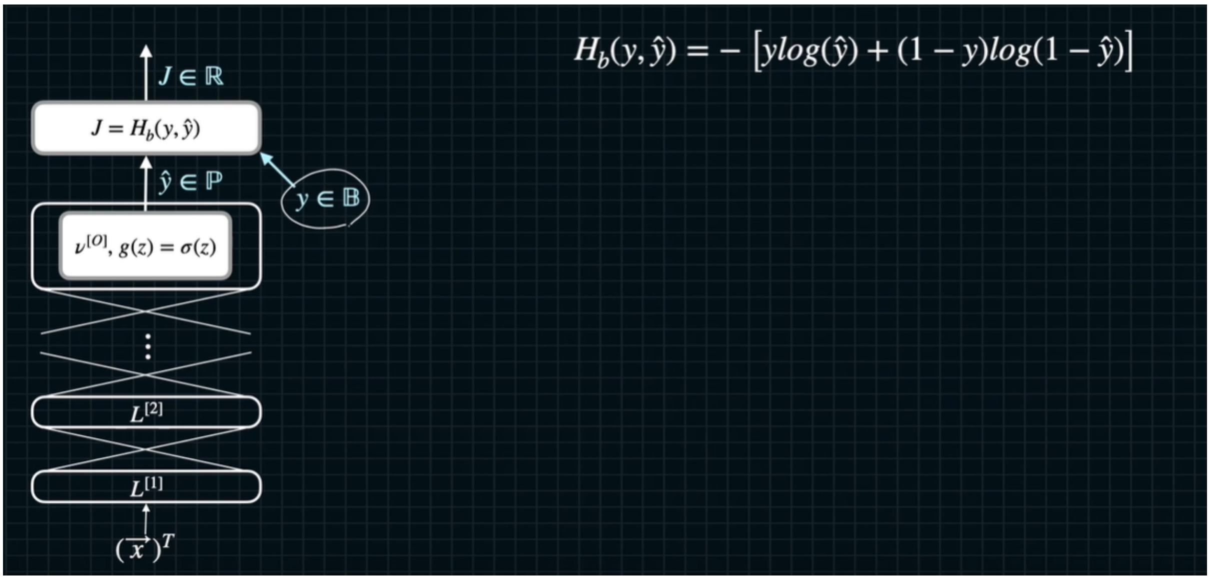
- 그래서 실제값이 1일때는 Loss의 형태는 -log(y^)값이되고 실제값이 0일 때는 Loss의 형태는 - log(1-y^)형태로 나온다







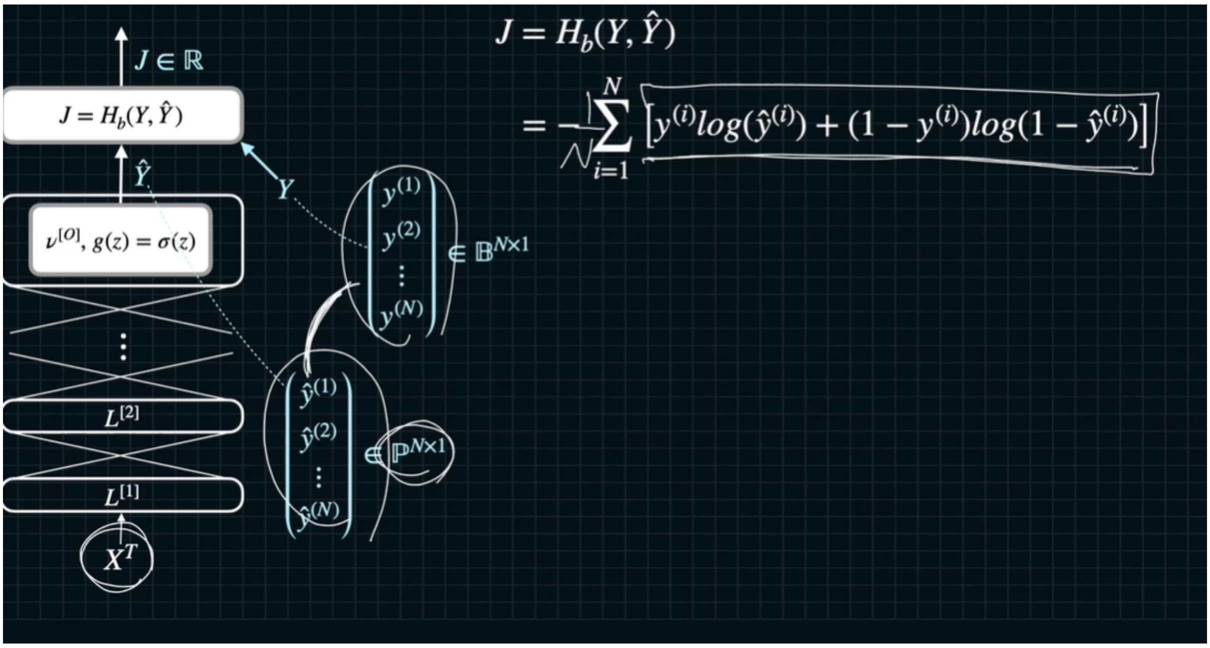
- 그래서 위 식은 y가 0일때 1일때 모두 사용할 수 있는 식이된다.



- 데이터 안에는 0, 1만들어 있음 그리하여 아웃풋은 결과는 0과 1사이의 값인 확률로 가지게 된다.
- x 데이터가 dense layer을 통과하면서 y^가 나오는걸 볼 수 있는데 자세히 보면 y^은 확률 값이다.확률값이 나온다. 그래서 우리는 마지막 layer에 activation function을 넣어주는거다.
- 그래서 우리는 sigmoid를 통과한 값들을 Binary Cross Entropy에 넣을 수 있게 되는 것이다.

- 그래서 우리는 뉴럴 네트워크를 만들 때 예를들어 고양이와 강아지를 분류하는 모델을 만들 때 마지막 output layer에는 sigmoid를 사용하면서 probablility를 출력을 하게 되는거고 이것을 가지고 Loss값을 구하게 되는것이다.

Minibatch일 때 Binary Cross Entropy형태



- 데이터를 미니베치형태로 넣을 때는 각각의 loss를 평균을 낸다고 생각하면 된다.