Loss (손실함수)와 Activate(활성화)함수

Loss함수 = 비용함수 (cost function)

즉, 손실로 인해 비용이 생긴다는 의미

그렇다면

실제값 y_set은 [0, 0, 1, 2, 1, 7]과 같이 주어지고
이를 0~5중 어떤 값을 나타내는 정확한 값이라고 하자.

(즉, 7일 확률이 7인 것.)

예측 예측값 ŷ_set을 [0.01, 0.07, 0.1, 0.2, 0.1, 0.3 등이라고 하자
=> 이는 계산된 예측된 데이터

그렇다면 우리는 예측값과 실제값의 오차를 줄여나갈 필요가 있고,
이때, 예측값과 실제값의 오차가
= 손실함수의 값이다.

손실함수에는 ┌ 평균제곱오차 (Mean Squared Error)
│ = MSE라고
└ 교차엔트로피오차 (Cross Entropy Error)
= CEE가 사용된다.

① MSE (평균제곱오차)

$$E = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

즉 Error는 예측값에서 실제값을 뺀 것의 제곱들을 더한 것의 평균입니다.

※파이썬에서 numpy.sum((ŷ - y)**2) 과 같은 수식으로 MSE를 쉽게 계산가능

이 MSE의 값이 클수록, 예측이 실제에서 크게 벗어났다는 것을 파악가능합니다.

* Neural Network 모델에 적용해서 표현하면

cost값 $J(W, b; x, y) = \frac{1}{2}\frac{1}{n}\sum_x (\hat{y}(x) - y)^2$ 이고

($\frac{1}{2}$을 미분시에 제곱 편함을 위함)

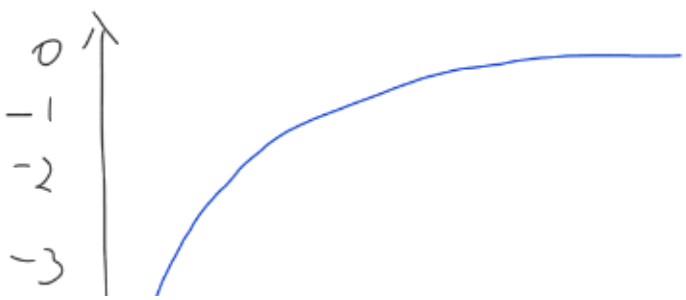
※ MSE의 장점은 거리차를 제곱하므로 차이가 적게나는 부분과 많이나는 부분이 뚜렷하게 구별되어 어느 part에서 오차가 두드러지는지를 파악가능.

↳ 분류문제에서 one-hot encoding 경우 사용하는 계산법.

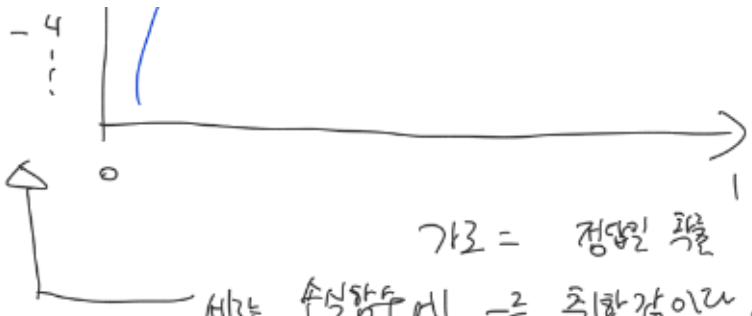
② CEE (교차엔트로피 오차) 입니다.

$$E = -\sum_{i=1}^{n} y_i \log \hat{y}_i \qquad (\hat{y}: 예측,\ y: 실제)$$

예측값에 밑이 e인 log를 취하고 실제값을 곱한 것들의 합에 − (음수)를 취합니다.



실제로 $\hat{y}_i$가 0이되면 −∞가 되버리므로 아주작은값 delta δ을 더해 방지한다.



가로 = 정답일 확률

세로 손실함수에 -를 취한 값이다. 즉, 정답에 가까워
질수록 y값은 0에 가까워진다.

CEE를 이용한 Cost function $J = -\frac{1}{n}\sum_x [y \log \hat{y} + (1-y)\log(1-\hat{y})]$

Q) 왜 MSE를 쓰고 분류문제에서 0~1 사이출력
// CEE를 쓸까?? 즉 sigmoid를 activation-function의
사용과 e함수 ∴ local minimum 안나타남
이때는 (EE 즉, ln함수사용이
효과적)

질문) CEE에서
왜 '(-)'를 취할까

질문) 이렇게 MSE와 CEE로 오차를 줄여나가서, 실제 y와
거의같은 값을 얻지?

⇒ Cost $J(w, b; x, y)$를 가지고 미분을 통해
최적의 w, b를 찾아 신경망을 업데이트시켜
줌으로써 Loss 함수값을 줄여나가 실제와근접하게
만들어나간다용

질문) 그럼 Loss 함수를 언제 써야하니?

⇒ y와 우리 accuracy를 출력해보다가
너무 정확도가 떨어지면 stop하고
Loss 함수를 계산하고 → cost function J를 미분해

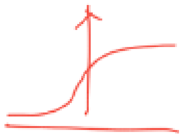
최적 w, b를 해 신경망을 갱신하고 다시 학습을 통해 비교해볼
필.

질문) 뭐가 더 좋은가?

* 신경망을 학습시키려면
  이 비용 J를 미분하여 최적의 기울기를 찾아야한다.

그리고 그때의 w, b를 얻어
  신경망에 '업데이트' 시켜줘야 한다.

문제는 MSE를 이용해 w, b를 업데이트시킬때
activation function이 sigmoid 일경우
z값이 너무 크거나 작으면 기울기가 0에 가까워서
잘못된 w, b를 개선하게되어 학습 저하 (slowdown
발생이 가능



$$\sigma(z) = \frac{1}{1+e^{-x}}$$

$\sigma(z)$는 sigmoid function

대신 CE를 사용하면

$$J = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

↓ 미분

$$J' = -\frac{1}{n}\sum_x \frac{\sigma'(z) x_{ij}}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y)$$

대입

$$J' w_{ij} = \frac{1}{n}\sum_x x_{ij}(\sigma(z)-y)$$

$$J' b \ldots = \frac{1}{n}\sum (\sigma(z)-y)$$

즉, Sigmoid에서 CEE를 쓰면 미분값이 예측값 $\sigma(z)$
실제값(y)의 차이에 비례함.

∴ 오차가 더 크면 더 많이 update
　 " 작으면 적게 " 하면되는 결과를 얻을수 있다.

그러므로 CEE가 MSE보다는 Neural network를 학습시키는데 더 적합한 Loss function 이라고 한다.

손실함수는 모든 train data에 적용해서 평균을 구해야 하는데 그러면 그 작업시간이 dataset 많을수록 오래걸림

↓

따라서 dataset의 일부만 랜덤으로 뽑아서 전체의 근사치로 사용한다.

이렇게 훈련 data의 일부를 추려 적용하는 것을 Mini-batch 라고 한다.

① Loss 함수가 왜 필요한가?

⇒ 예측값이 잘못나왔을때 신경망을 제대로 학습시키위한 지표로써

손실함수는 연속적인 값을 리턴가능하다.

(0000개 중 9000개를 맞추면 0.9 뽑아내는
0001개를 맞추면 0-9001

0-900125등의
연속적인 값에 대한 오차계산이
가능하다.

그러나 정확도는 불연속적으로 변화하므로
매번이 불가
그럴 초점화가 힘들다.

∴ 연속적으로 변화하는 초점함수를 지표로 사용하여 확증

마지막 수정: 오후 8:04