Neural Networks의 학습속도 저하(Slowdown)를 막는 Cross-Entropy Cost Function

2017년 11월 12일 by Solaris

머신러닝(Machine Learning) 알고리즘을 학습시키기 위해서 일반적으로 다음의 과정을 커친다.

1. Cost Function을 정의한다.
2. Gradient Descent 알고리즘을 이용해서 파라미터를 업데이트한다.

이때 Cost Function으로 널리 쓰이는 형태 중 하나는 **Mean Squared Error(MSE) Cost Function**이다. MSE Function은 모델의 예측값(Prediction)-$\hat{Y}$-과 실제 타겟값(True target value)-Y-과의 차이를 제곱해서 모두 더한 값들의 평균으로 정의된다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{1}$$

이를 Neural Networks의 모델에 적용해서 표현하면 Neural Networks에서 output unit에서의 Cost function-$J(W, b; x, y)$-은 모델의 예측값(Prediction)-$\sigma(z)$-과 실제 타겟값(True target value)-y-과의 차이를 제곱한 값으로 정의된다. (참고 : $z = Wx + b$, 계산의 편의성을 위해서 $\frac{1}{2}$을 식에 추가함)

$$J(W, b; x, y) = \frac{1}{2}\frac{1}{n}\sum_{x}(\sigma(z) - y)^2 \tag{2}$$

Neural Networks를 학습시키기 위해서는 Cost function-$J(W, b; x, y)$-의 미분값(derivative)를 계산해서 Neural Networks의 파라미터 W, b를 업데이트해야한다.MSE Cost function을 이용했을 경우 Weight와 Bias의 미분값(Derivative)를 계산하면 아래와 같다.

$$\frac{\partial}{\partial W_{ij}}J(W, b; x, y) = \frac{1}{n}\sum_{x}(\sigma(x) - y)\sigma'(z)x_{ij} \tag{3}$$

$$\frac{\partial}{\partial b_{ij}}J(W, b; x, y) = \frac{1}{n}\sum_{x}(\sigma(x) - y)\sigma'(z) \tag{4}$$

이때 문제점은 sigmoid function을 activation function-σ-으로 쓸 경우, z값이 너무 크거나 작아지면 미분값-$\sigma'(z)$-이 급격히 0에 가까워지는 현상이 발생한다. 따라서 이는 학습속도가 저하(slowdown)되는 문제를 유발하게 된다.

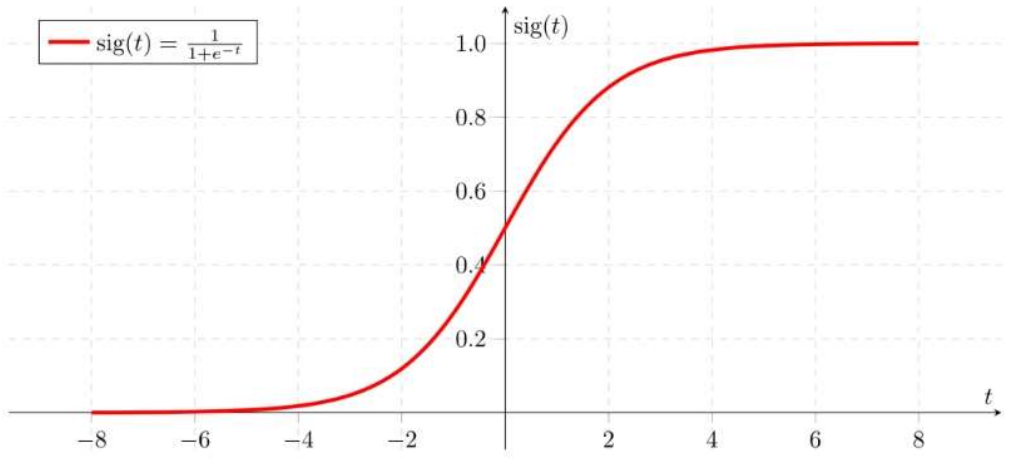

그림 1 – sigmoid function (그림에서 약 -6이하, 6이상인 구간에서 미분값-$\sigma'(z)$-이 0에 가까워진다.)

이런 문제 때문에 Neural Networks를 학습시키기 위해 MSE Cost Function 대신 **Cross-Entropy Cost Function**을 더 활발하게 사용하고 있다. Cross-Entropy를 이용한 Cost Function-$J(W,b;x,y)$-은 아래와 같이 정의된다.

$$J(W,b;x,y) = -\frac{1}{n}\sum_{x}[ylna + (1-y)ln(1-a)] \tag{5}$$

Cross-Entropy Cost Function을 이용했을 때 Neural Networks의 파라미터인 Weight의 미분값(Derivative)를 계산하면 아래와 같다.

$$\frac{\partial}{\partial W_{ij}}J(W,b;x,y) = -\frac{1}{n}\sum_{x}(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)})\sigma'(z)x_{ij} \tag{6}$$

위 식을 전개해서 다시 정리하면 아래와 같다.

$$\frac{\partial}{\partial W_{ij}}J(W,b;x,y) = -\frac{1}{n}\sum_{x}\frac{\sigma'(z)x_{ij}}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y) \tag{7}$$

sigmoid function은 $\sigma(z) = \frac{1}{(1+e^{-z})}$로 정의되고 이를 미분하면 $\sigma'(z) = \sigma(z)(1-\sigma(z))$이다. 따라서 이를 이용해 위 식을 다시한번 정리하면 최종적으로 Cross-Entropy를 이용한 Cost Function-$J(W,b;x,y)$-의 Weight의 미분값(Derivative)은 아래와 같이 계산된다.

$$\frac{\partial}{\partial W_{ij}}J(W,b;x,y) = \frac{1}{n}\sum_{x}x_{ij}(\sigma(z)-y) \tag{8}$$

같은 방식으로 Bias의 미분값을 계산하면 아래와 같다.

$$\frac{\partial}{\partial b_{ij}}J(W,b;x,y) = \frac{1}{n}\sum_{x}(\sigma(z)-y) \tag{9}$$

위 식에서 파라미터 Weight와 Bias의 미분값(Derivative)은 **예측값과 실제값의 차이-$(\sigma(z)-y)$-에 비례한다. 따라서 오차가 더큰 인풋값에 대해서는 더많이 업데이트하고, 오차가 더 작은 인풋값에 대해서는 더적게 업데이트하는 결과를 얻을 수 있다.**

또한, 미분값에 $\sigma'(z)$가 포함되어 있지 않기 때문에 MSE Function을 이용해서 Cost Function을 정의했을때 발생하는 sigmoid function의 특성때문에 발생하는 학습저하(slowdown) 문제도 발생하지 않는다.

따라서 Cross-Entropy Cost Function이 MSE Cost Function보다 Neural Networks를 학습시키기는데 더 적합한 Cost Function임을 알 수 있다.

References

[1] http://neuralnetworksanddeeplearning.com/chap3.html