인공지능

- 신경 회로망 상세 설명 및 실험 (Cost Function 1) -

부산대학교

LG Electronics

목차

인공지능 개요 (신경 회로망 주요 연산)

I. 신경 회로망 주요 연산 1
(cost 함수)

I. 신경회로망 주요 연산 1

Cost 함수 의미 및 특성

참고자료 - http://neuralnetworksanddeeplearning.com/chap3.html

신경회로망 주요 연산

- Neural network 학습을 위한 기법
  - Cost function (Error function)
    - MSE(Mean Squared Error)
    - Cross entropy function

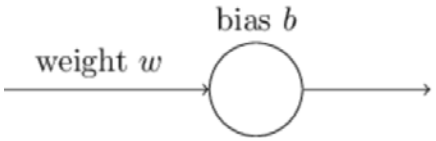
Cost Function - MSE

신경회로망 초기값 설정에 따른 학습 특성 비교(1/2)

– 입력이 '1'일때 '0'을 출력하는 neural network 학습하는 경우

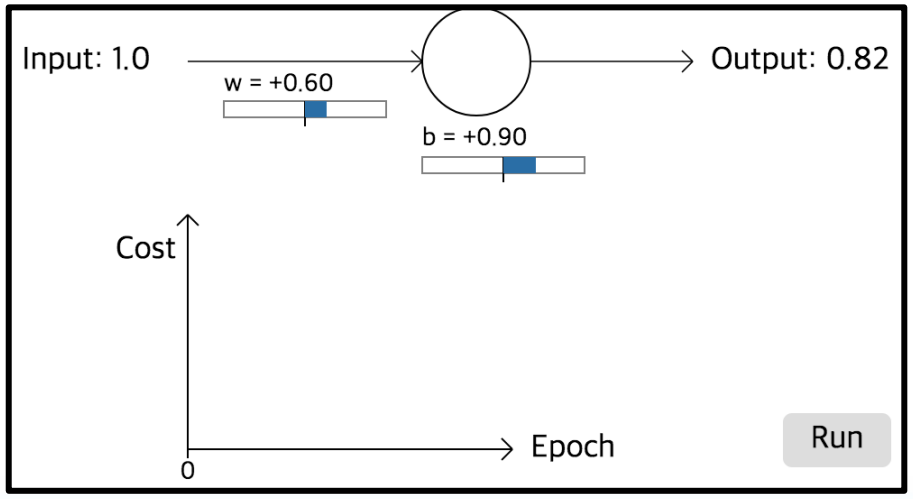
- "inversion"을 수행하는 신경 회로를 학습시키자 !
- weight w와 b 학습
- 한편, cost function은 quadratic cost를 사용하는 경우임!

$$C = \frac{(y-a)^2}{2},$$

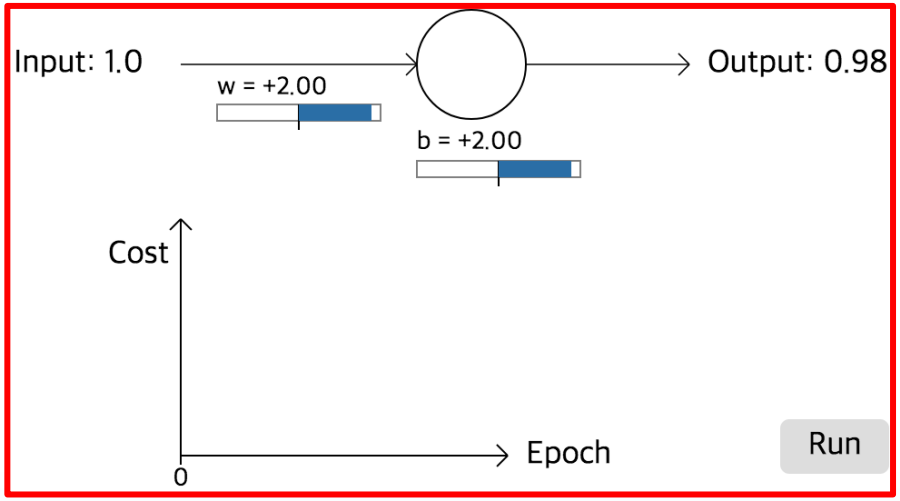


– Weight w와 bias b 초기값에 따른 학습 속도 비교(학습율 η = 0.15로 동일하게 설정)

(1) 초기 w=0.6, 0.9일 경우의 학습 특성?



(2) 초기 w=2.0, 2.0일 경우의 학습 특성?

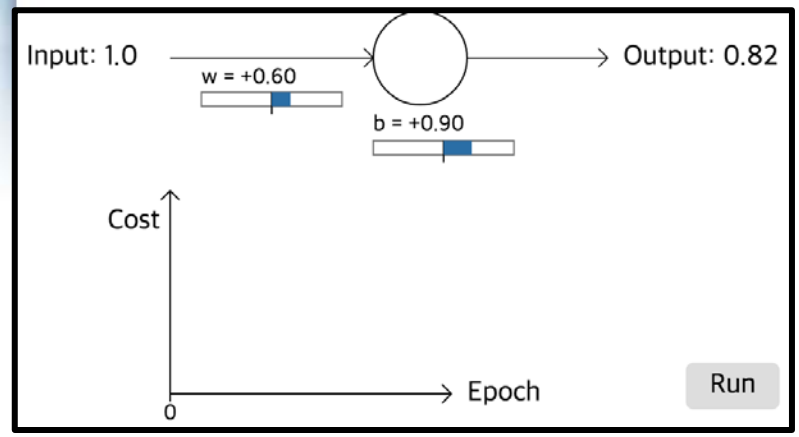


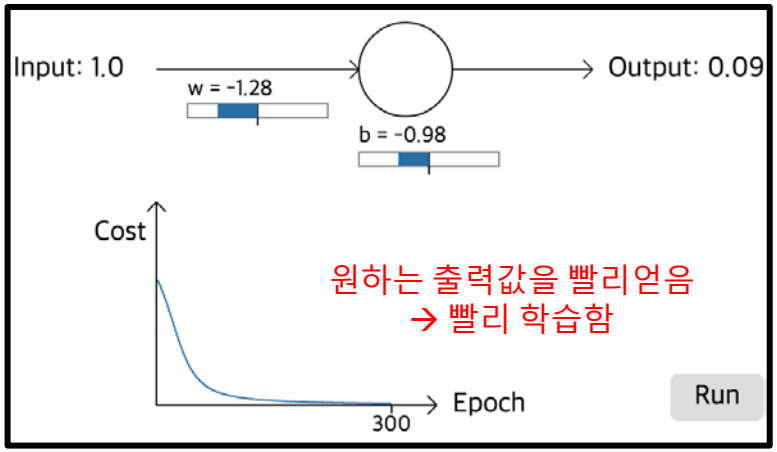
http://neuralnetworksanddeeplearning.com/chap3.html

Cost Function - MSE

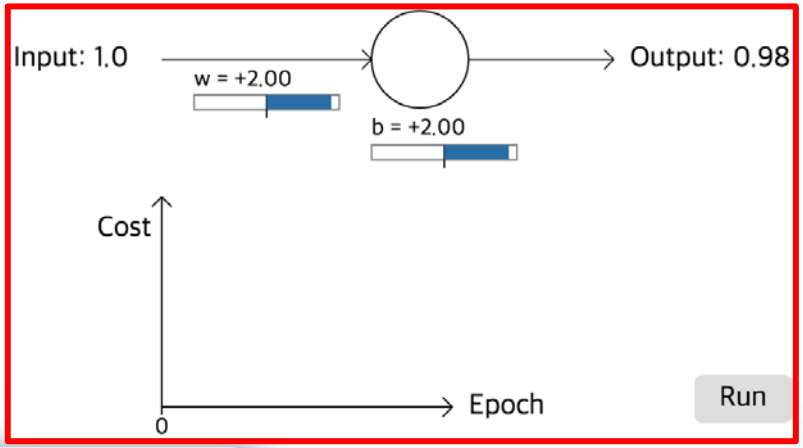
Inversion을 수행 neural network의 초기값 설정에 따른 학습 특성 비교(2/2)

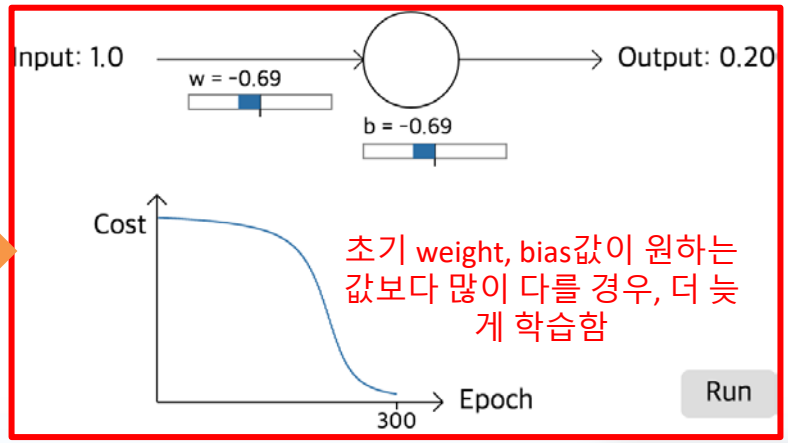
(1) 초기 w=0.6, b=0.9일때의 학습 특성?





(2) 초기 w=2.0, b=2.0일때의 학습 특성?

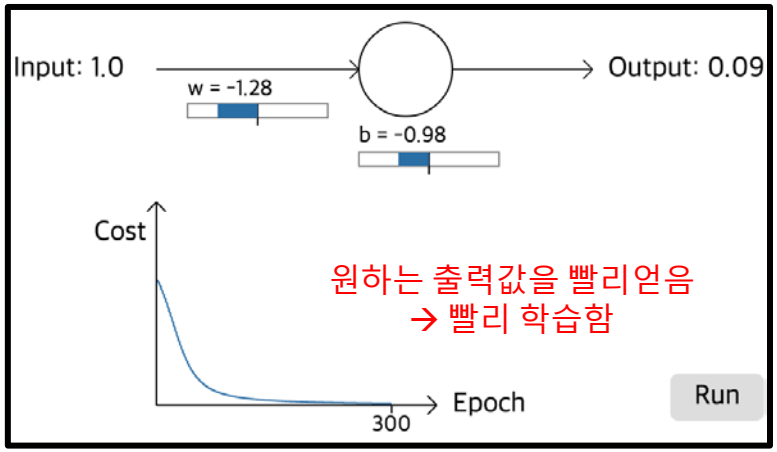




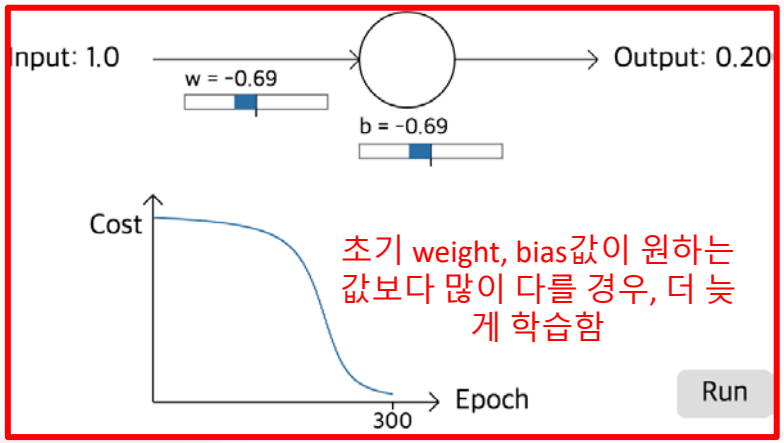
Cost Function - MSE

Inversion을 수행 neural network의 초기값 설정에 따른 학습 특성 비교(2/2)

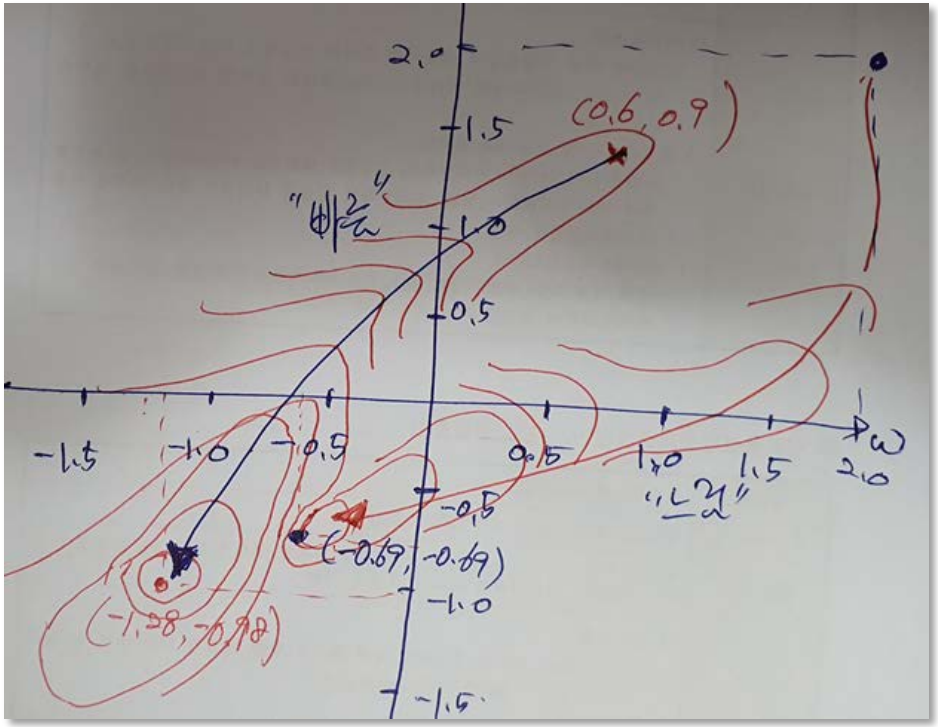
(1) 초기 w=0.6, 0.9일 경우의 학습 특성?



(2) 초기 w=2.0, 2.0일 경우의 학습 특성?



W,b의 초기값이 적은 경우 ((1) 0,6, 0.9)가 많은 경우 ((2), 2.0, 2.0)보다 더 빨리 낮은 cost로 수렴함



Cost Function - MSE

- **왜 w, b의 초기 값이 적은 경우가 많은 경우보다, 더 빨리 학습하는가?**
  - 이는 사람의 일반적인 학습 패턴과는 다름 (사람은 실수를 할 수록 더 빨리 배우지만, 신경 회로망은 원하는 값과 많이 다르면 더 늦게 배우게 됨)
  - 신경망에서는 Parameter에 대한 Cost 함수의 미분값에 의해, w와 b의 변화값이 결정됨

$$C = \frac{(y-a)^2}{2}$$

비용 함수

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z) \tag{55}$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z), \tag{56}$$

w와 b에 대한 Cost 편미분

해당 neuron은 Inversion logic이므로,
- x=1일 때, y=0
- x=0일 때, y=1이 되어야 함

$$a = \sigma(z)$$

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

Sigmoid 함수

$$z = wx + b$$

Neuron node의 입력값

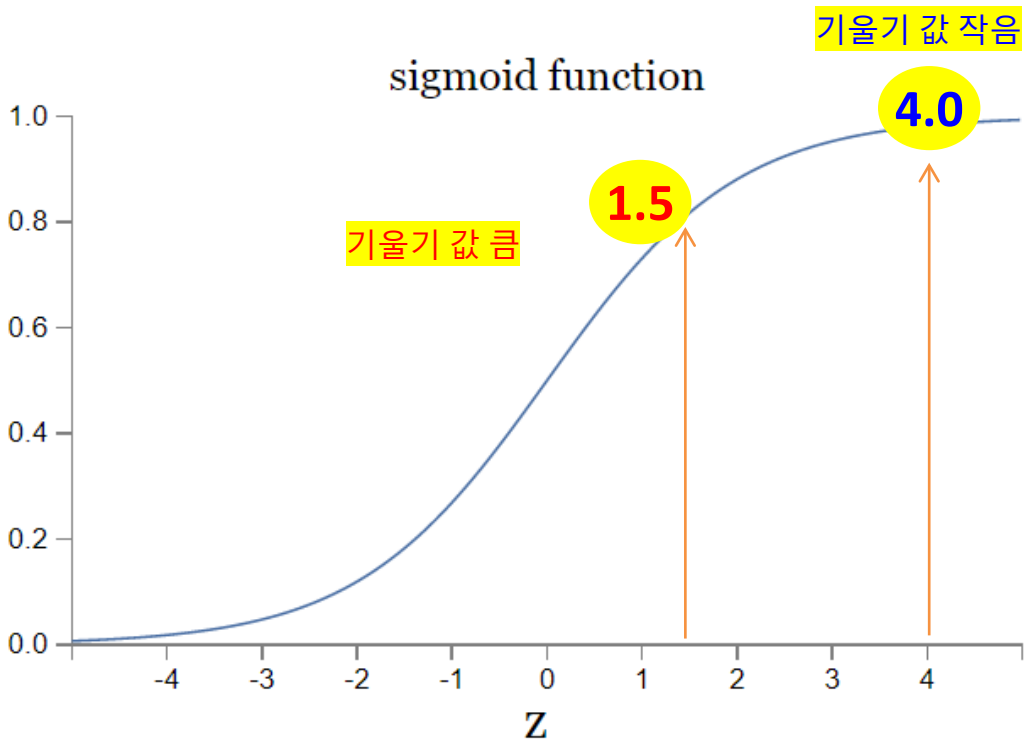
  - 위 수식 (55)와 (56)번을 보면, neuron의 학습은 $\sigma'(z)$에 의해 결정됨을 알 수 있음

Cost Function - MSE

Neuron의 학습은 $\sigma'(z)$에 의해 결정됨을 알 수 있음

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z) \qquad (55)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z), \qquad (56)$$



여기서, neuron의 학습은 $\sigma'(z)$에 의해 결정된다고 했음. sigmoid 함수는 아래처럼 입력 x에 대해 weight와 bias에 의한 z 값임

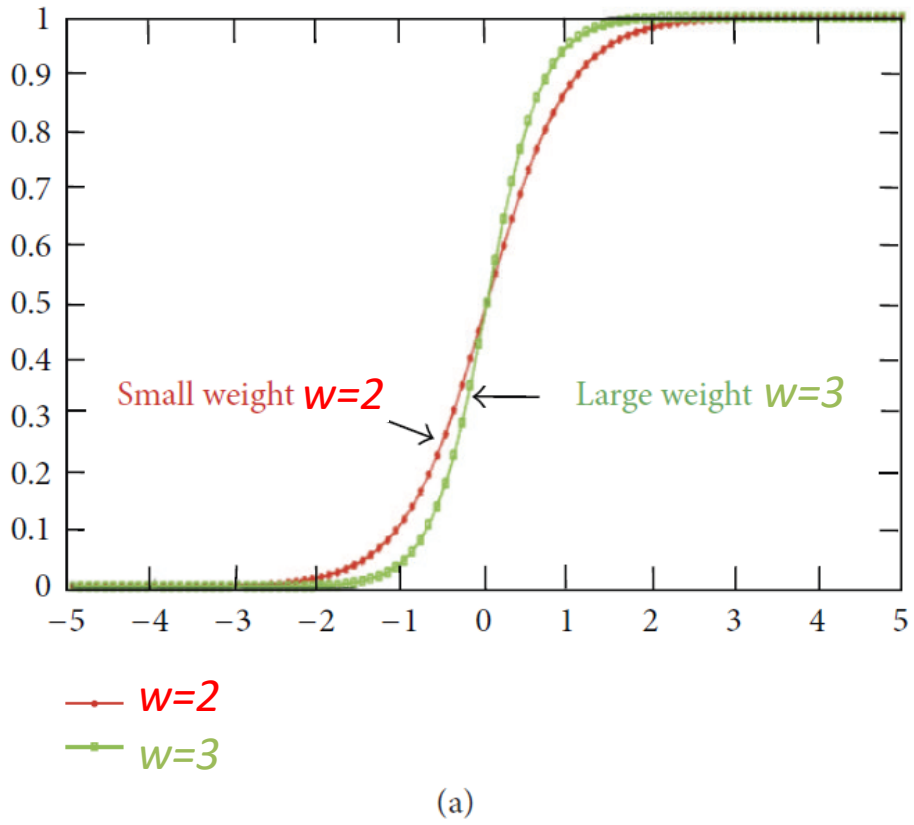
$$z = wx + b$$

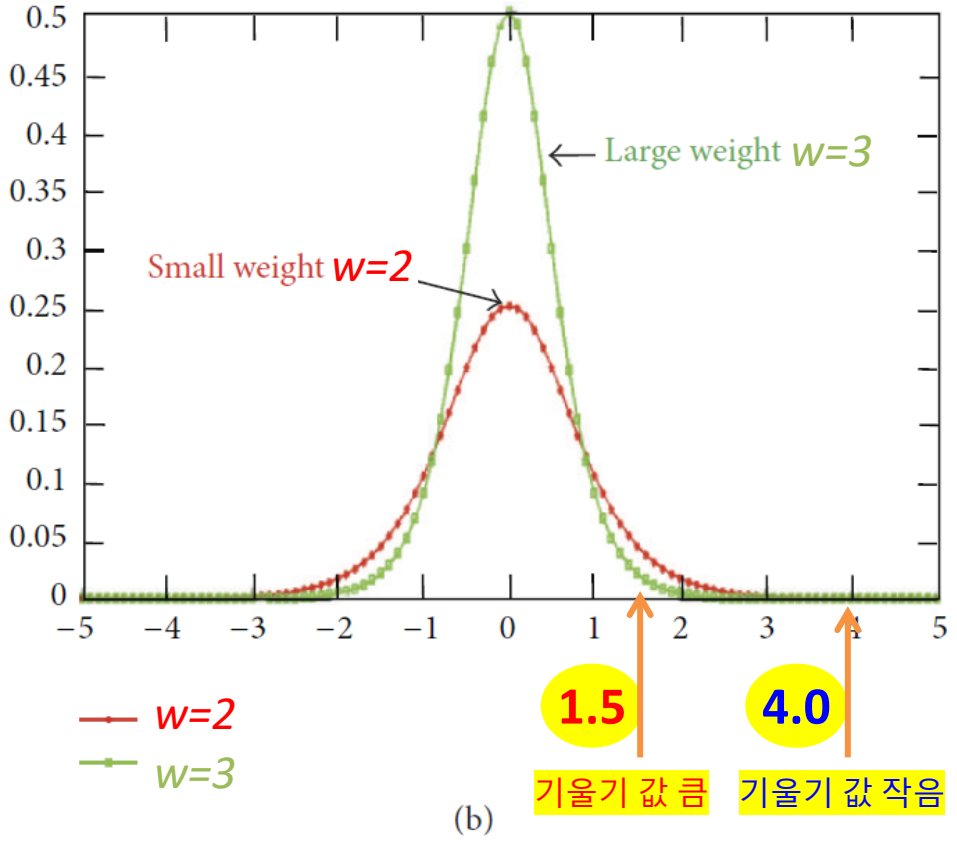
- W와 b의 값이 작으면, x=1일 때, z는 작은값
- 미분한 값은 큰값
- w와 b의 값이 크면, x=1일 때, z는 큰 값. 미분한 값은 작은값

(1) W= 0.6, b=0.9일 때, z=0.6 * 1 + 0.9 = 1.5
(2) W=2.0, b=2.0일 때, z=2.0*1 + 2.0=4.0

Cost Function - MSE

- W의 값에 따른 Sigmoid 함수와 sigmoid 함수의 미분값(derivative)의 변화





즉, w와 b의 값이 크면, x=1일때, z는 큰 값이며, 이때의 $\sigma'(z)$ 값은 작은 값임을 알 수 있음. 즉, 천천히 학습함 !!

Cost Function - MSE

- Cost 함수와 parameter에대한 미분

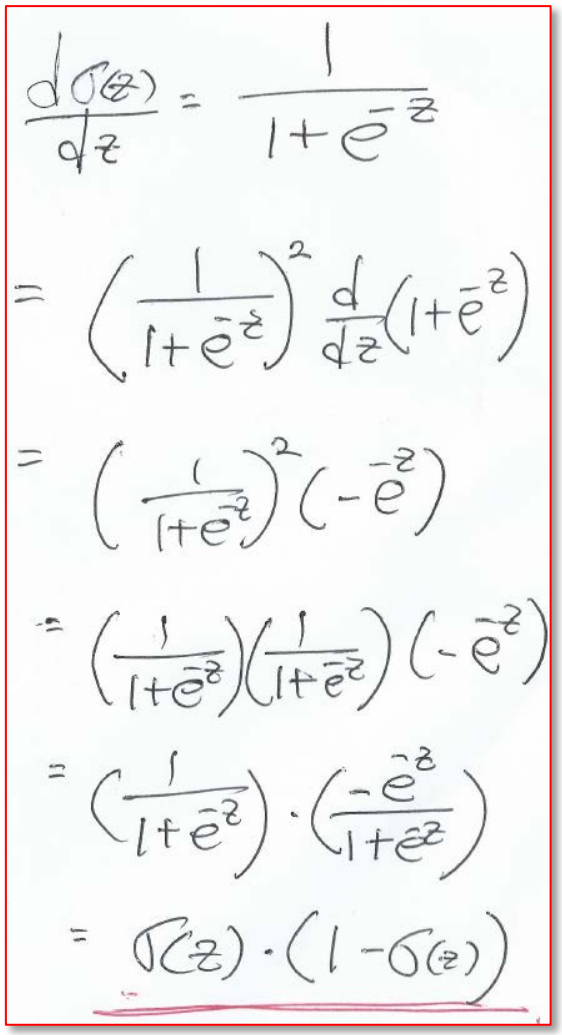
$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z),$$

- Derivative of the sigmoid

$$\boldsymbol{\sigma'(z) = \frac{d\,\sigma(z)}{dz} =}$$



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

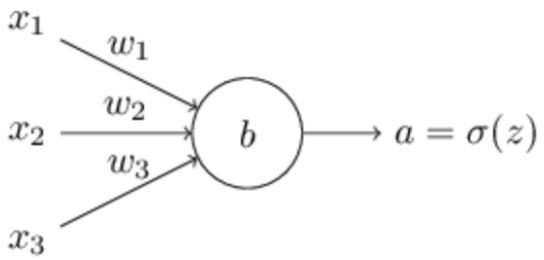
$$\frac{d\,\sigma(z)}{dw} = \frac{d\,\sigma(z)}{dz} * \frac{dz}{dw}$$

$$= w\,\sigma(z)(1-\sigma(z))$$

Cross-entropy cost function

quadratic 대신, cross-entropy 를 cost 함수로 사용하여, slow down 해결

different cost function, known as the cross-entropy. To understand the cross-entropy, let's move a little away from our super-simple toy model. We'll suppose instead that we're trying to train a neuron with several input variables, $x_1, x_2, \ldots$, corresponding weights $w_1, w_2, \ldots$, and a bias, b:



The output from the neuron is, of course, $a = \sigma(z)$, where $z = \sum_j w_j x_j + b$ is the weighted sum of the inputs. We define the cross-entropy cost function for this neuron by

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)], \tag{57}$$

where n is the total number of items of training data, the sum is over all training inputs, x, and y is the corresponding desired output.

Cross-entropy cost function

see this, let's compute the partial derivative of the cross-entropy cost with respect to the weights. We substitute $a = \sigma(z)$ into (57), and apply the chain rule twice, obtaining:

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j} \tag{58}$$

$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j. \tag{59}$$

Putting everything over a common denominator and simplifying this becomes:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y). \tag{60}$$

Using the definition of the sigmoid function, $\sigma(z) = 1/(1+e^{-z})$, and a little algebra we can show that $\sigma'(z) = \sigma(z)(1-\sigma(z))$. I'll ask you to verify this in an exercise below, but for now let's accept it as given.

Cross-entropy cost function

We see that the $\sigma'(z)$ and $\sigma(z)(1-\sigma(z))$ terms cancel in the equation just above, and it simplifies to become:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j (\sigma(z) - y). \tag{61}$$

This is a beautiful expression. It tells us that the rate at which the weight learns is controlled by $\sigma(z) - y$, i.e., by the error in the output. The larger the error, the faster the neuron will learn. This is just what we'd intuitively expect. In particular, it avoids the learning slowdown caused by the $\sigma'(z)$ term in the analogous equation for the quadratic cost, Equation (55). When we use the cross-entropy, the $\sigma'(z)$ term gets canceled out, and we no longer need worry about it being small. This cancellation is the special miracle ensured by the cross-entropy cost function. Actually, it's not really a miracle. As

Cross-entropy cost function

the cross-entropy cost function. Actually, it's not really a miracle. As we'll see later, the cross-entropy was specially chosen to have just this property.

In a similar way, we can compute the partial derivative for the bias. I won't go through all the details again, but you can easily verify that

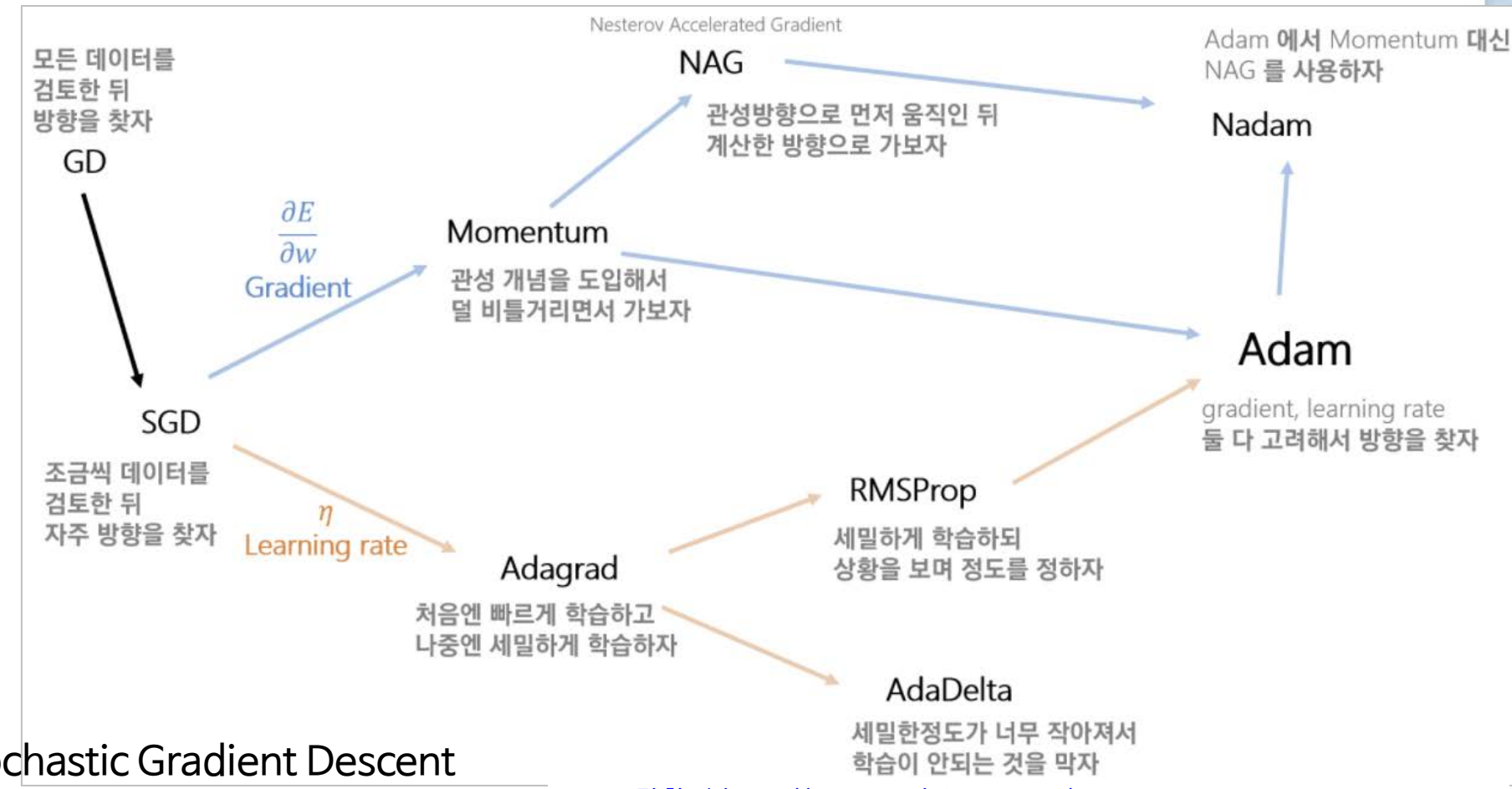
$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y). \tag{62}$$

Again, this avoids the learning slowdown caused by the $\sigma'(z)$ term in the analogous equation for the quadratic cost, Equation (56).

최적화 기법

머신러닝 기법(신경 회로망)을 위한 최적화 기법(Optimizer) 종류

- Gradient Descent와 여러 Optimizer



Stochastic Gradient Descent

Nesterov Accelerated Gradient

: 그림 참고) https://gomguard.tistory.com/187

: https://keras.io/ko/optimizers/#adagrad