Ch12.

다층 인공신경망을 밑바닥부터 구현

202STG18 이재빈

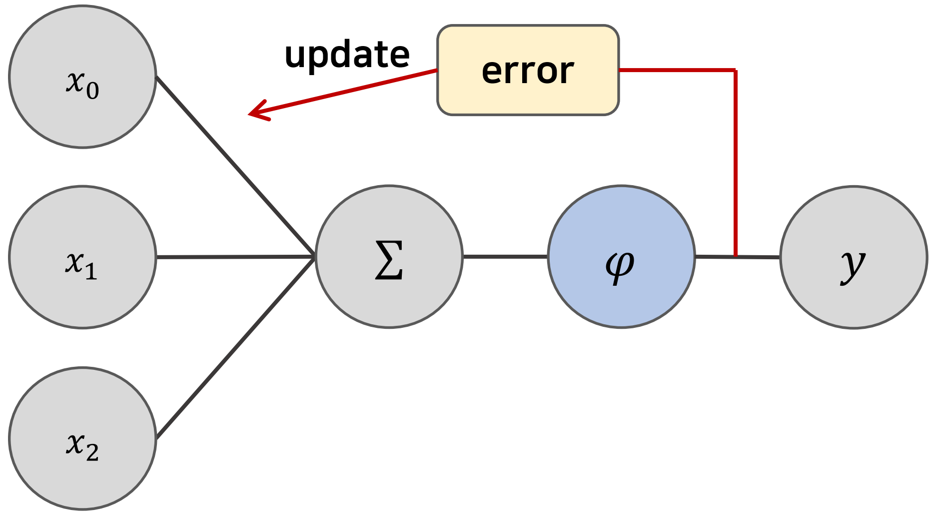
CONTENTS

01 ,

Neural Network

1. Neural Network

- Neural Net의 역사 : 1세대



Perceptron

1. **Rosenblatt, F. (1958)**

   The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain

   가장 단순한 형태의 계산에 의한 최초 신경망 모델

2. **Widrow, B. & Hoff, M. (1960)**

   Adaptive Switching Circuits

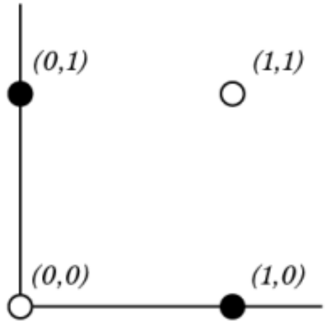
   Adaline, 오차에 따라 가중치 갱신하는 신경망 모델

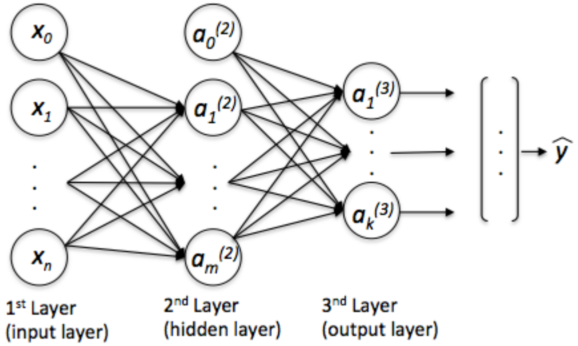
1. Neural Network

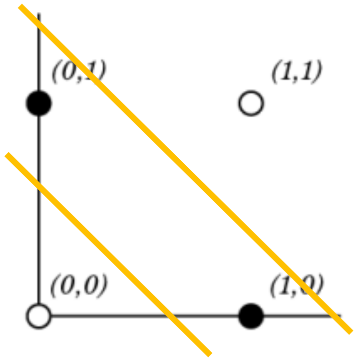
- Neural Net의 역사 : 1세대 -> 2세대

4. XOR GATE

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0







단층 퍼셉트론은 XOR 문제를 해결할 수 없다

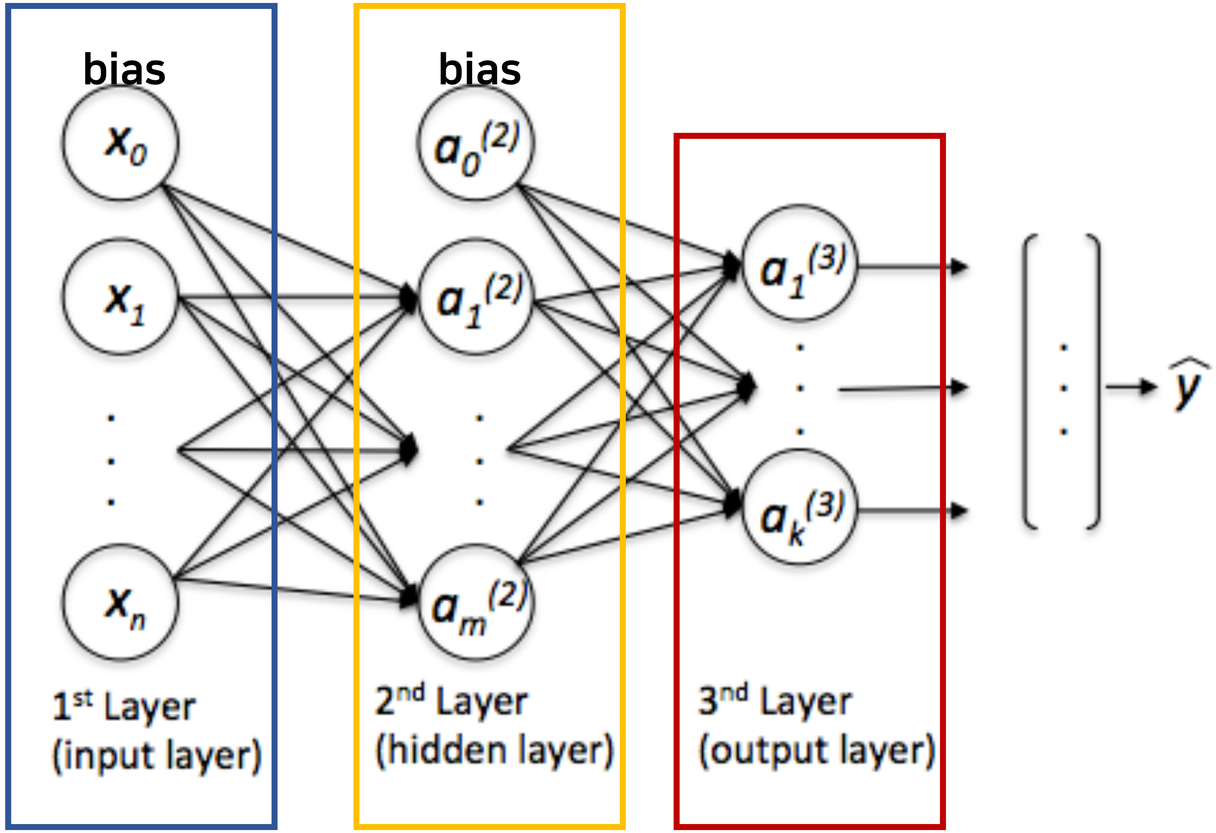
Minsky, M. & Papert, S. (1969)
Perceptrons : an introduction
to computational geometry

Hidden layer 를 추가시킨 다층 퍼셉트론
XOR 문제 해결 가능!
이를 학습시키는 오류 역전파 방법

David E. Rumelhart, Geoffrey E. Hinton
& Ronald J. Williams (1986)
Learning representations by back-propagating errors

1. Neural Network

- Multilayer Perceptron 의 구조



(n+1) 개의 입력 노드

(m+1) 개의 은닉층 노드

k 개의 출력 노드

Parameter 개수

(n+1)*m + (m+1)*k 개의 weight

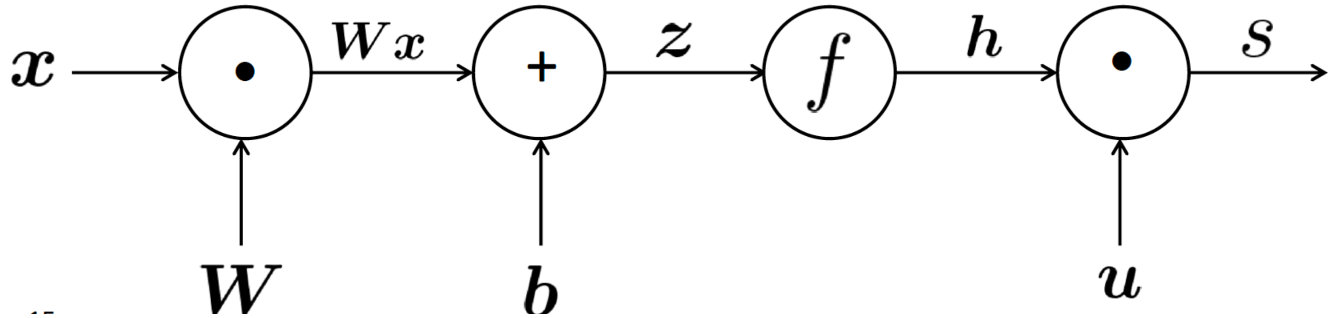
각각의 파라미터 값들을 모두 업데이트!

02 ,

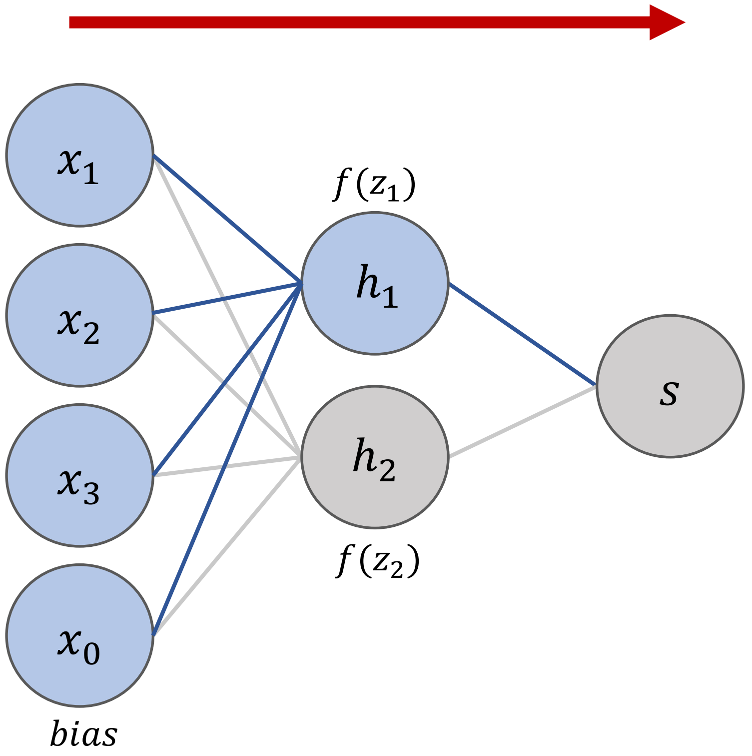
FeedForward & BackPropagation

2. FeedForward & BackPropagation
- FeedForward

Forward Propagation

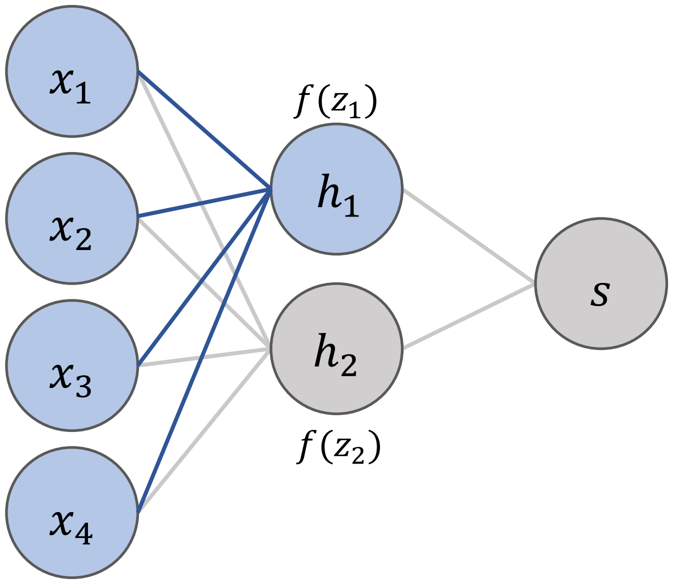


- Feedforward : 각 층에서 입력을 순환시키지 않고 다음 층으로 전달



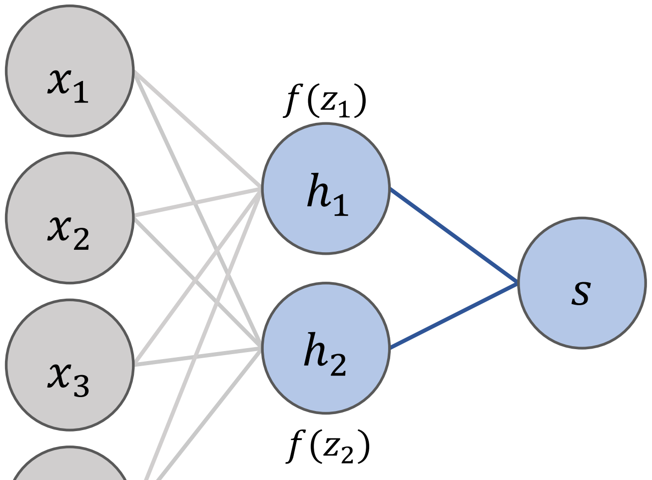
2. FeedForward & BackPropagation

- FeedForward



$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$z_1 = \sum_{k=1}^{4} w_{1k} x_k$$



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

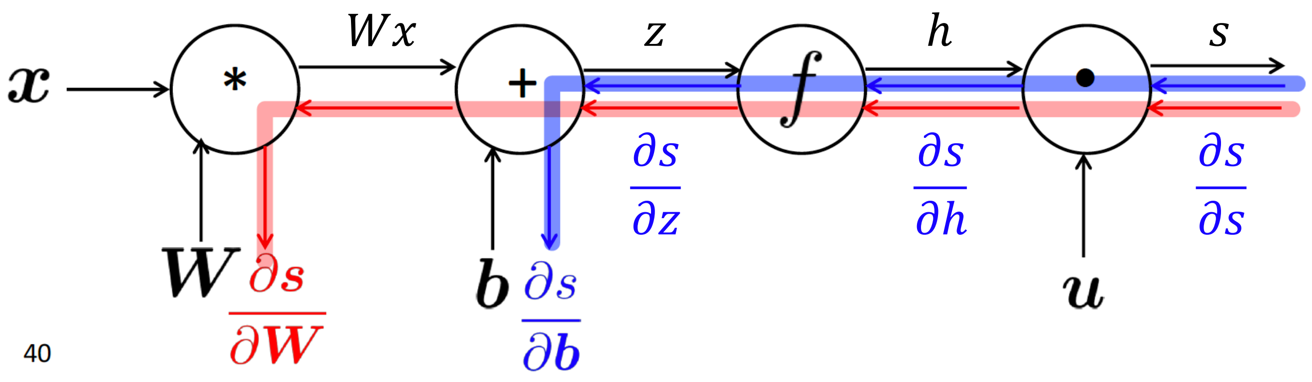
$$z = Wx$$

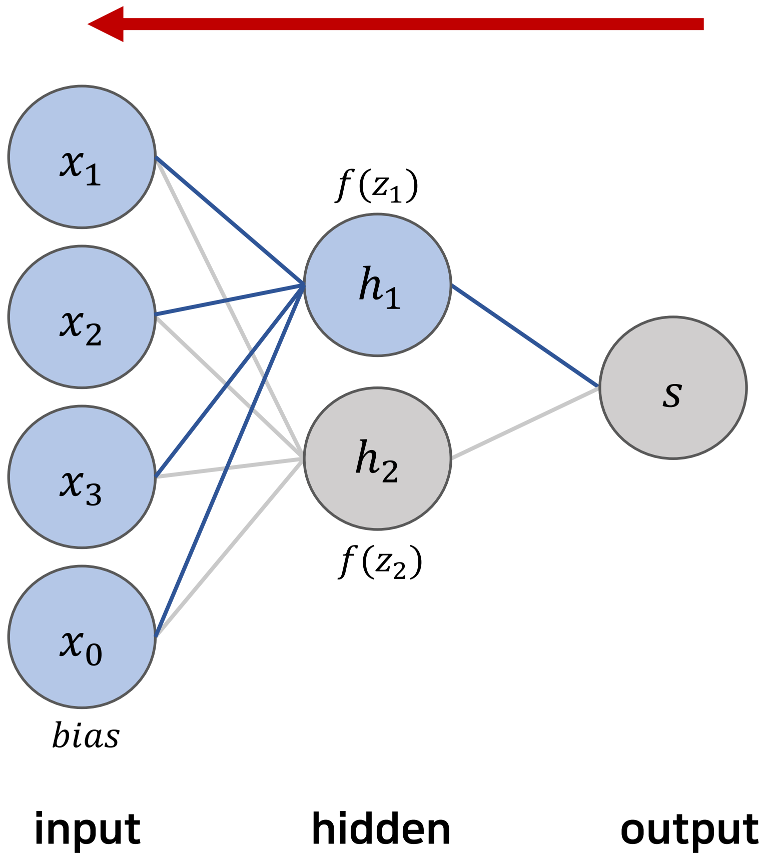
$$s = u^T z = u_1 z_1 + u_2 z_2$$

2. FeedForward & BackPropagation

- BackPropagation

Back Propagation

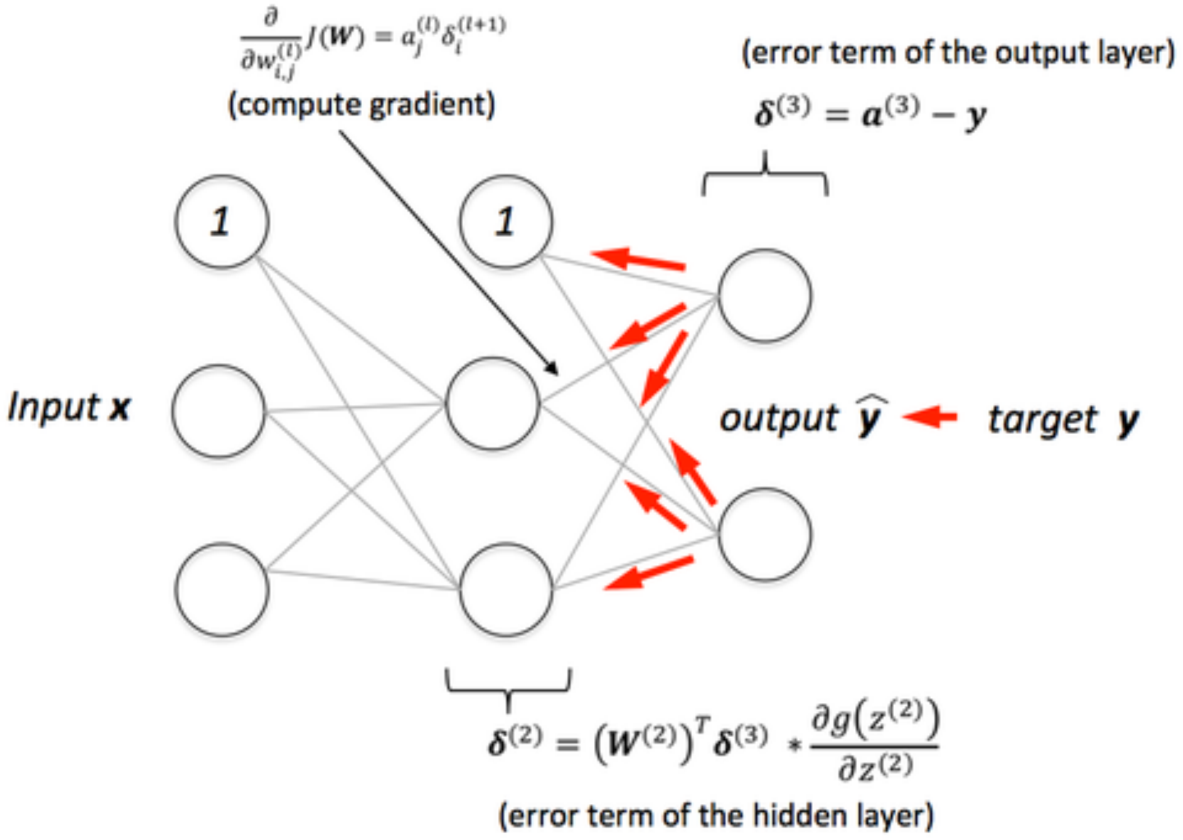




- 역방향 자동 미분의 특별한 경우 (오른쪽 → 왼쪽)
- 행렬과 벡터를 곱해서 또 다른 벡터를 얻은 후, 다음 행렬을 곱함
- 행렬-벡터 곱셈은 행렬-행렬 곱셈보다 훨씬 계산 비용이 적게 듦

- 계산했던 지난 과정들이 다시 사용됨으로써 다시 계산하여 계산량을 늘리는 문제를 막을 수 있음

2. FeedForward & BackPropagation
- Chain Rule



$$\boxed{\frac{\partial s}{\partial W} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial W}}$$

Weight matrix

Chain Rule

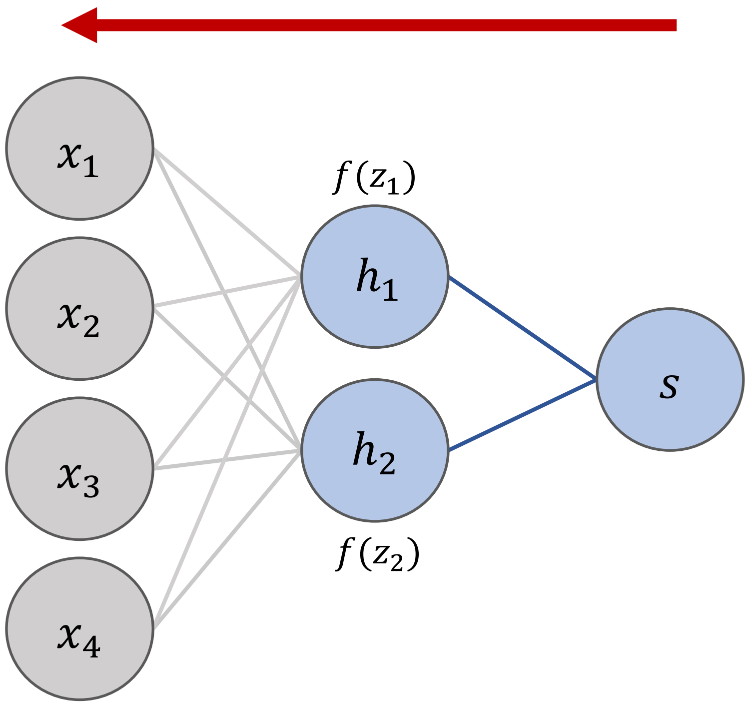
함수의 연쇄법칙

합성함수 미분

$$F = (f \circ g)(x) = f\big(g(x)\big)$$
$$F' = (f \circ g)'(x) = f'\big(g(x)\big)g'(x)$$

2. FeedForward & BackPropagation

- BackPropagation



$$z = Wx$$
$$s = u^T z = u_1 z_1 + u_2 z_2$$

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boxed{\frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} \rightarrow \delta = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}$$

$$\frac{\partial s}{\partial W_{ij}} = \delta \frac{\partial z}{\partial W_{ij}} = \sum_{k=1}^{4} \delta \frac{\partial z_k}{\partial W_{ij}} = \delta_i x_j$$

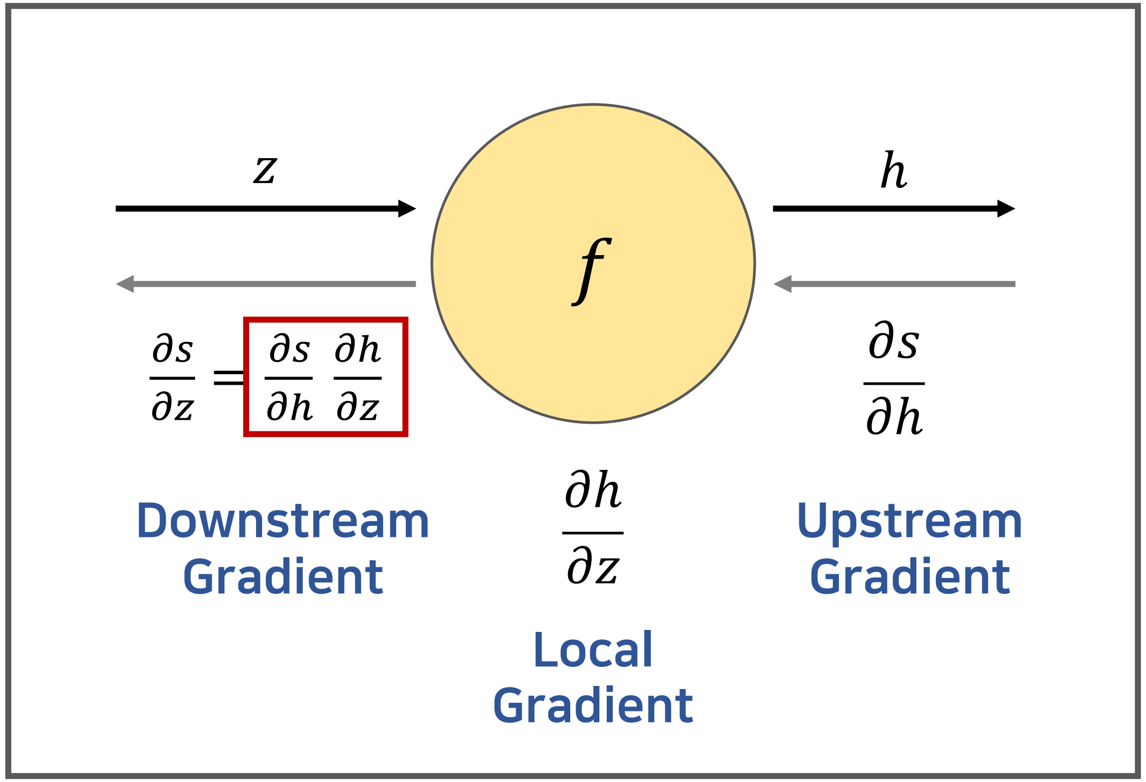
δ : Error signal from above
x : Local gradient signal

$$\frac{\partial s}{\partial W} = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} [x_1 \; x_2 \; x_3 \; x_4] = \delta x^T$$

m x n n x 1 1 x m

2. FeedForward & BackPropagation

- BackPropagation



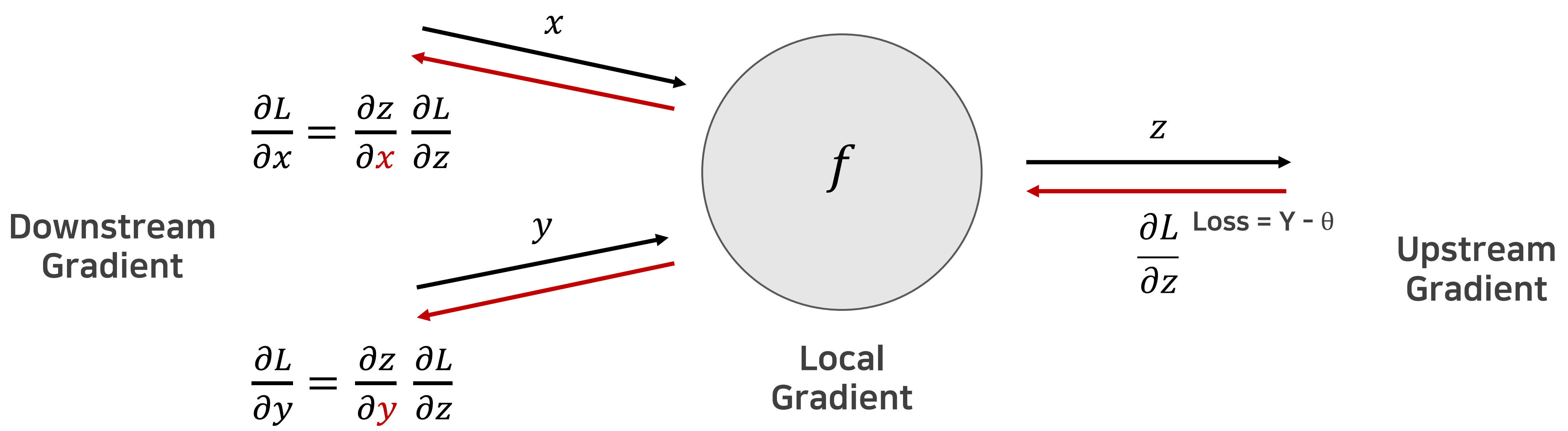
출처 : Stanford CS224n – Natural Language Processing with Deep Learning

1. **Local Gradient** = $\frac{\partial(output)}{\partial(input)}$

2. **Downstream Gradient**
   = Local Gradient * Upstream Gradient
   (∵ Chain Rule)

2. FeedForward & BackPropagation
- Gradient Flow

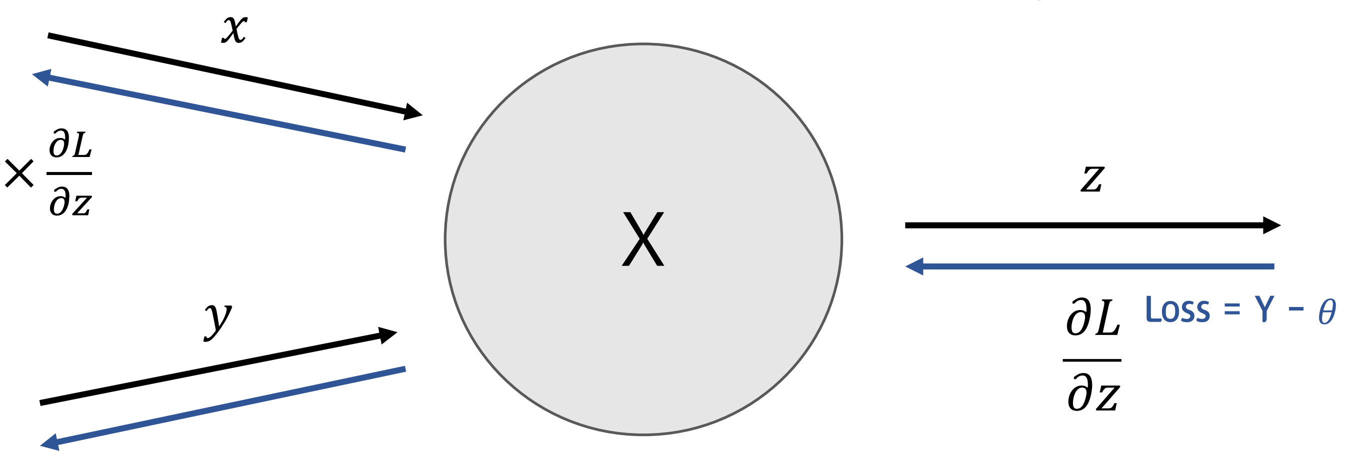
역전파 분해



2. FeedForward & BackPropagation
- Gradient Flow

곱셈의 역전파

$$z = f(x, y) = xy$$



$$\frac{\partial L}{\partial x} = y \times \frac{\partial L}{\partial z}$$

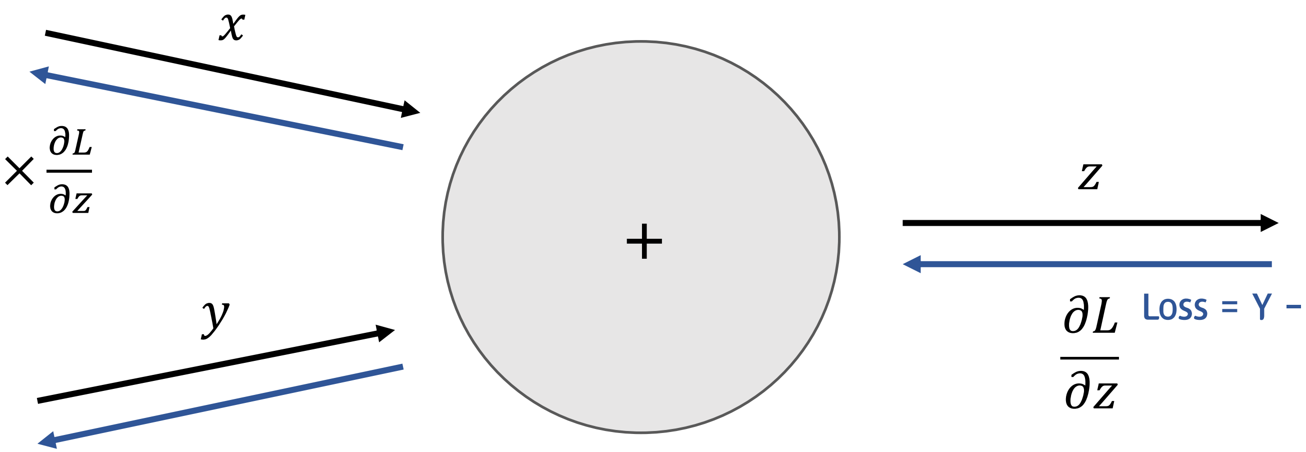
$$\frac{\partial L}{\partial y} = x \times \frac{\partial L}{\partial z}$$

2. FeedForward & BackPropagation

- Gradient Flow

덧셈의 역전파

$$z = f(x, y) = x + y$$



$$\frac{\partial L}{\partial x} = 1 \times \frac{\partial L}{\partial z}$$

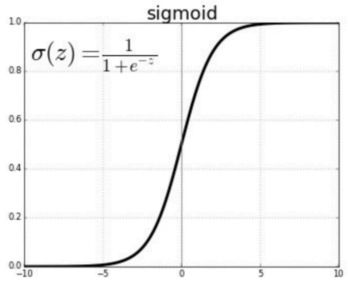
$$\frac{\partial L}{\partial y} = 1 \times \frac{\partial L}{\partial z}$$

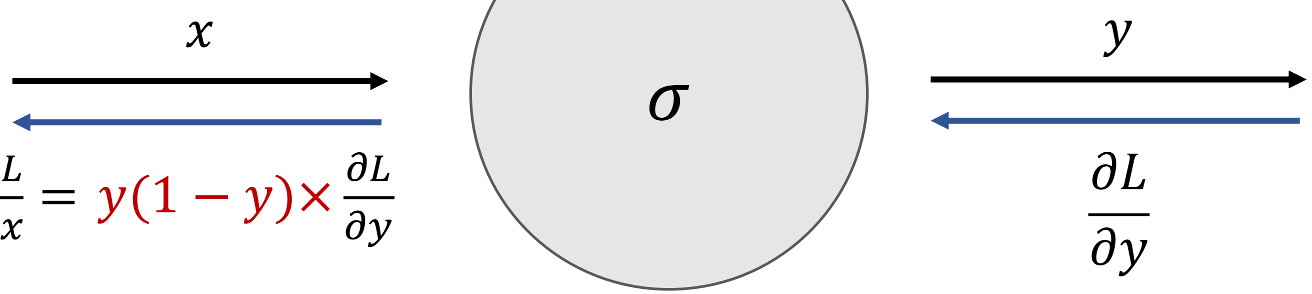
$$\frac{\partial L}{\partial z}$$ Loss = Y - θ

2. FeedForward & BackPropagation
- Gradient Flow

시그모이드 역전파

$$y = \frac{1}{1+e^{-x}}$$





$$\frac{\partial L}{\partial x} = {\color{red} y(1-y)\times} \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial y}$$

sigmoid 미분

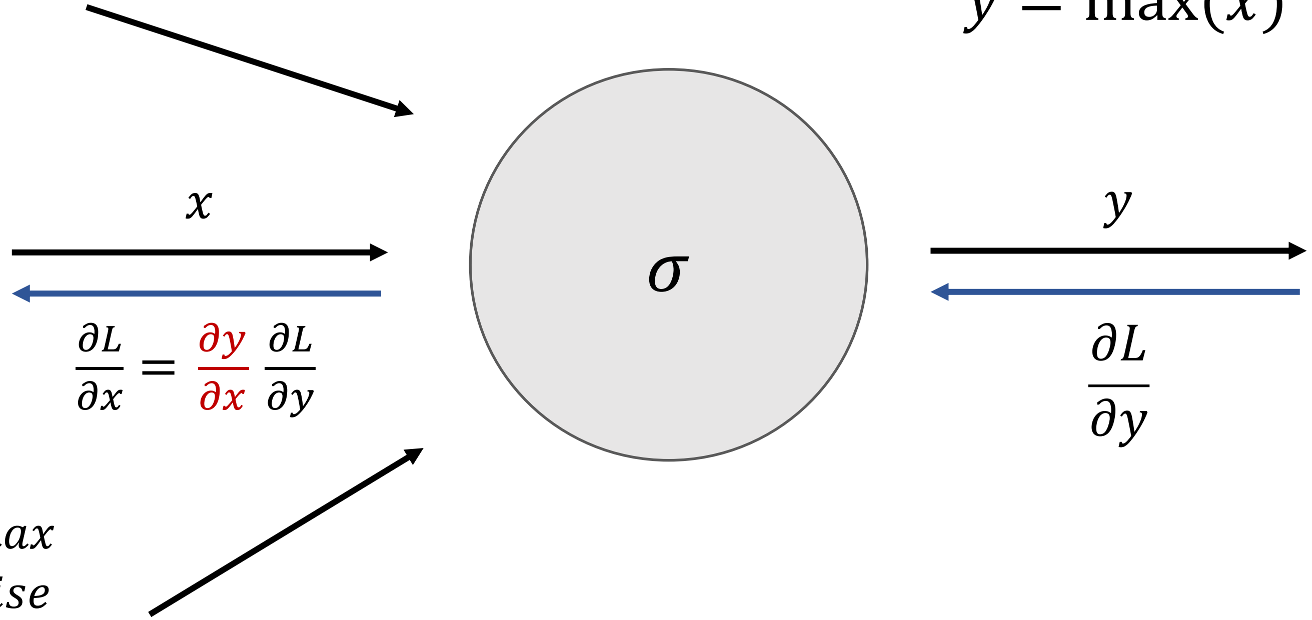
$$\frac{\partial \sigma}{\partial x} = \frac{-(-e^{-x})}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \times \frac{e^{-x}}{1+e^{-x}} = \sigma(x)\{1-\sigma(x)\}$$

2. FeedForward & BackPropagation

- Gradient Flow

max 역전파

$$y = \max(x)$$

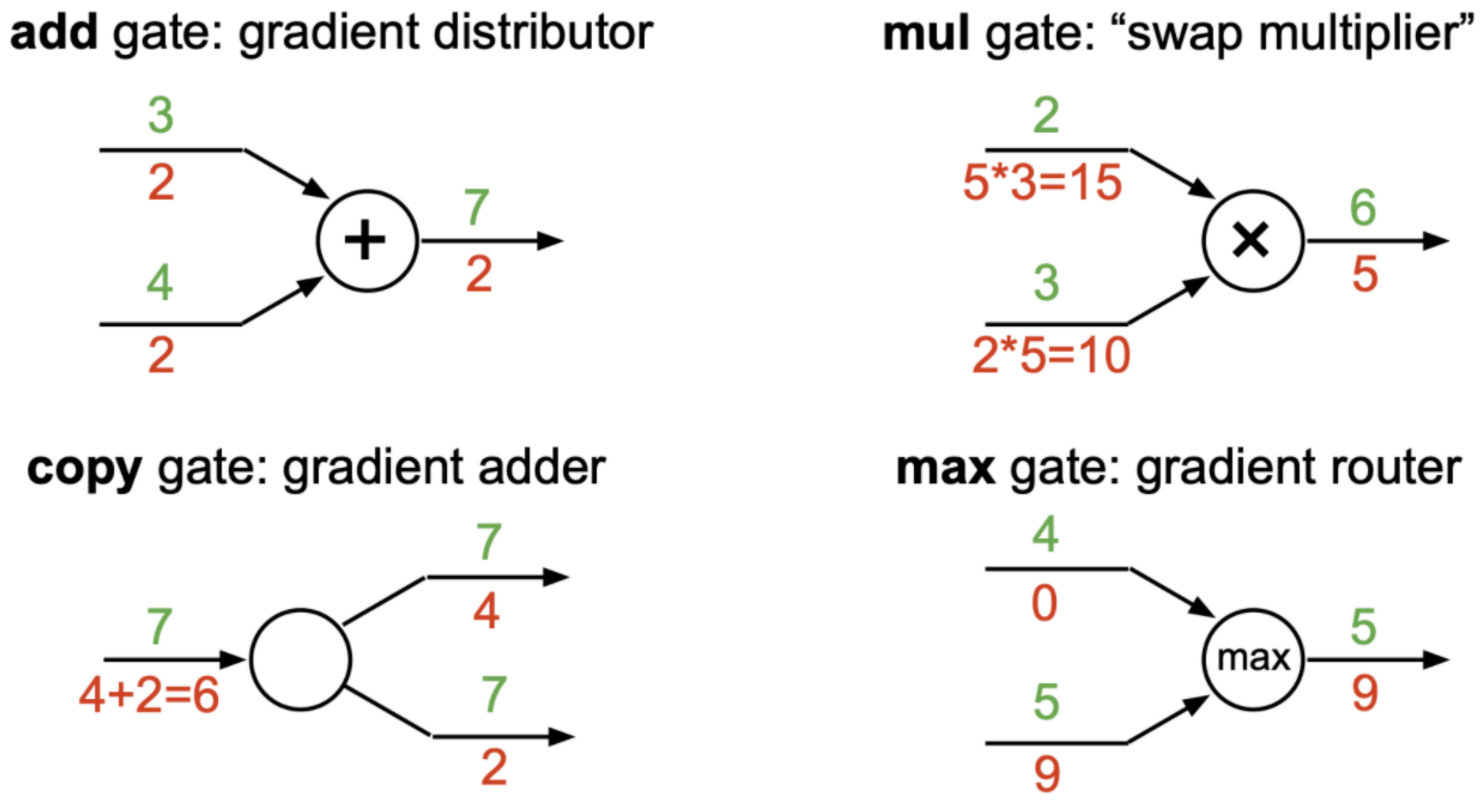


$$\frac{\partial L}{\partial x} = \frac{\partial y}{\partial x} \frac{\partial L}{\partial y}$$

$$\frac{\partial y}{\partial x} = \begin{cases} 1 \ if \ x \ is \ max \\ 0 \ \ otherwise \end{cases}$$

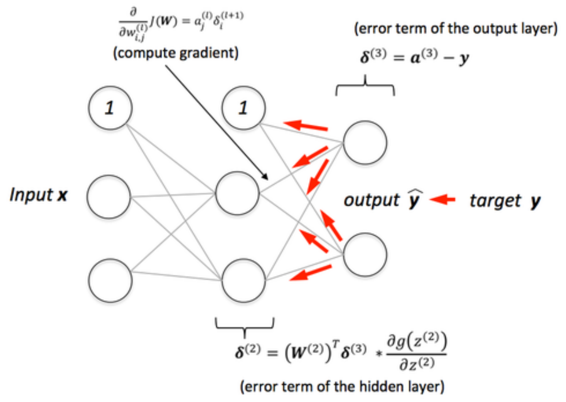
2. FeedForward & BackPropagation

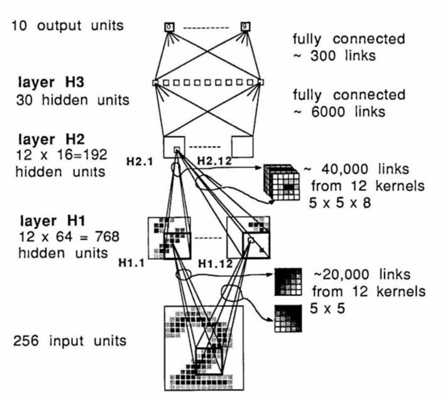
- Gradient Flow



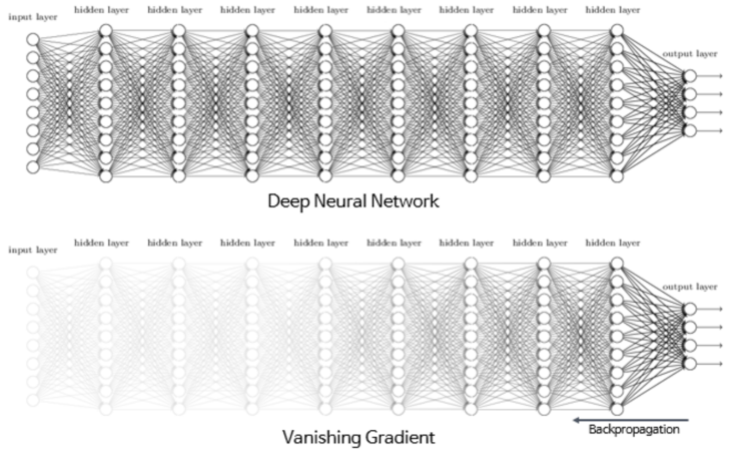
2. FeedForward & BackPropagation

- Neural Net의 역사 : 2세대





Backpropagation applied to handwritten zip code recognition,
Lecun et al., 1989



BackPropagation

David E. Rumelhart,
Geoffrey E. Hinton
& Ronald J. Williams (1986)
Learning representations by
back-propagating errors

Conv NeuralNet

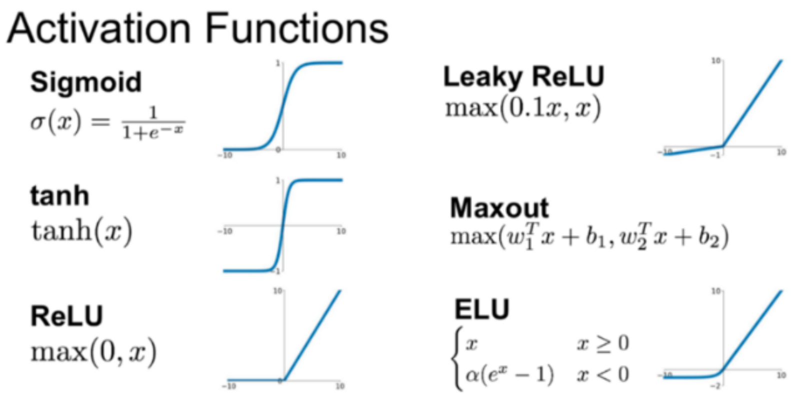
LeCun, Y. et al. (1989)
Backpropagation Applied to
Handwritten Zip Code
Recognition

Vanishing Gradient

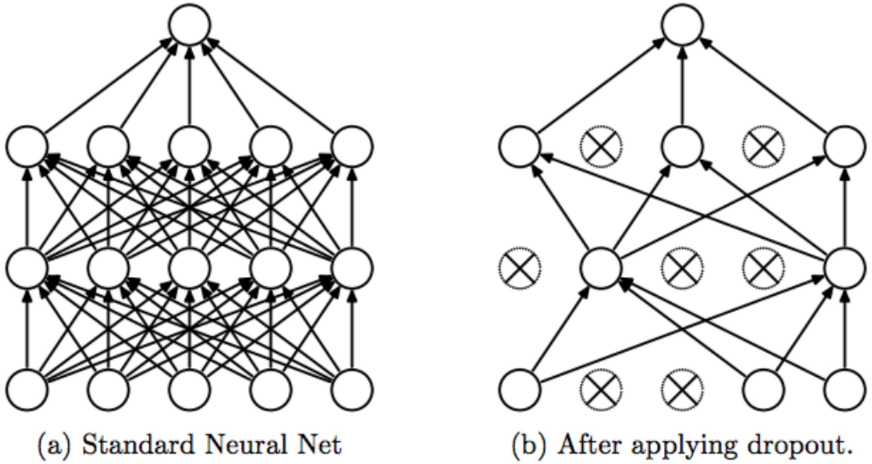
Y. Bengio, P. Simard
& P. Frasconi (1994)
Learning long-term dependencies
with gradient descent is difficult

2. FeedForward & BackPropagation

- Neural Net의 역사 : 3세대



Different Activation Functions and their Graphs



ReLU

Nair, V. & Hinton, G. E. (2010)
Rectified Linear Units Improve
Restricted Boltzmann Machines

Dropout

Geoffrey E. H, Nitish S , Alex K,
Ilya S & Ruslan R. S. (2012)
Improving neural networks by preventing
co-adaptation of feature detectors

03 ,

Code

감사합니다 😊