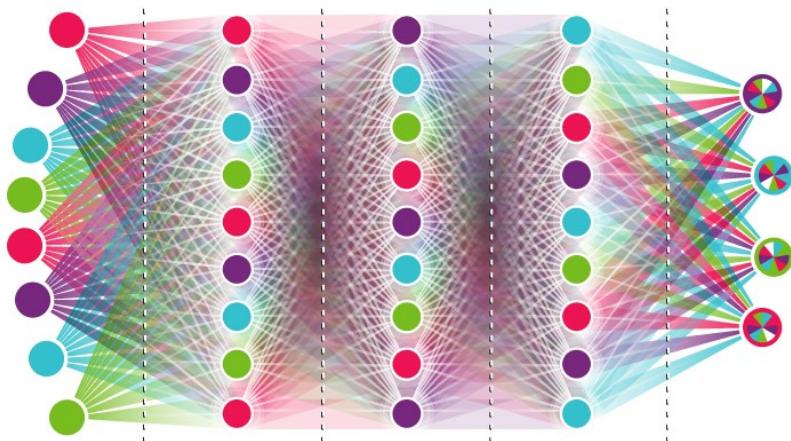


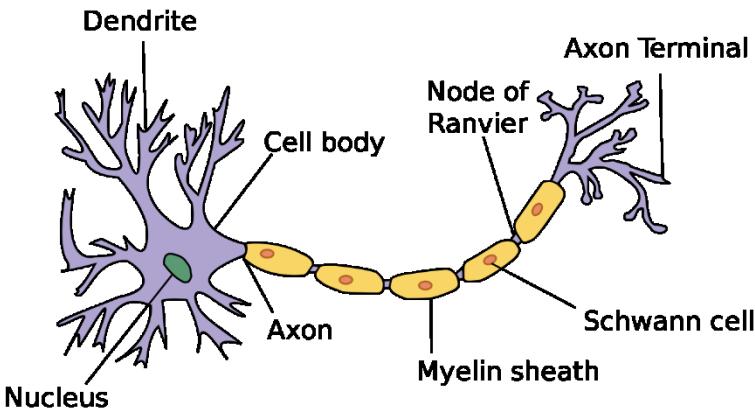
# Multi-Layer Perceptron

The Road to Deep Learning



Fast Campus  
Start Deep Learning with TensorFlow

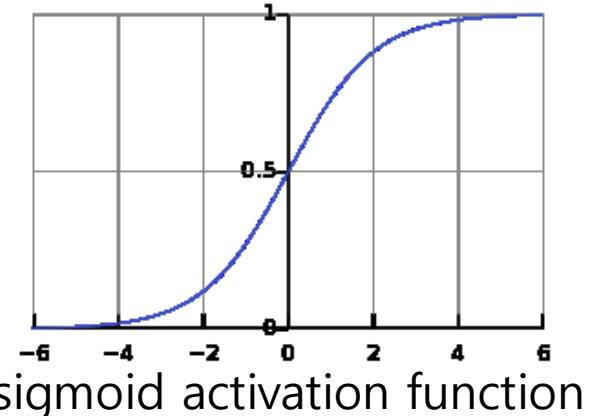
# Recap - Perceptron



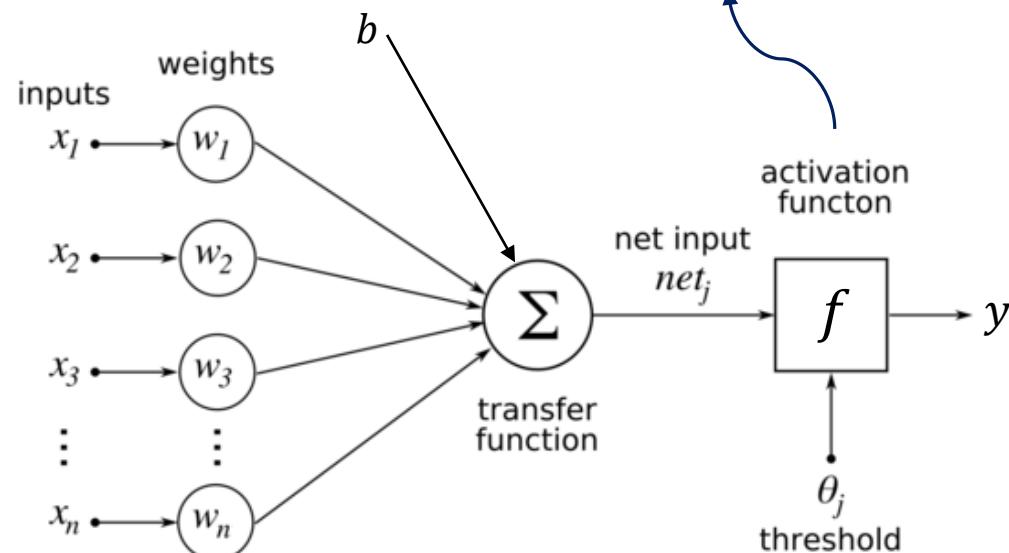
$$y = f(\mathbf{w}\mathbf{x} + b)$$

$$\mathbf{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$$

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ \dots \ x_n]^T$$

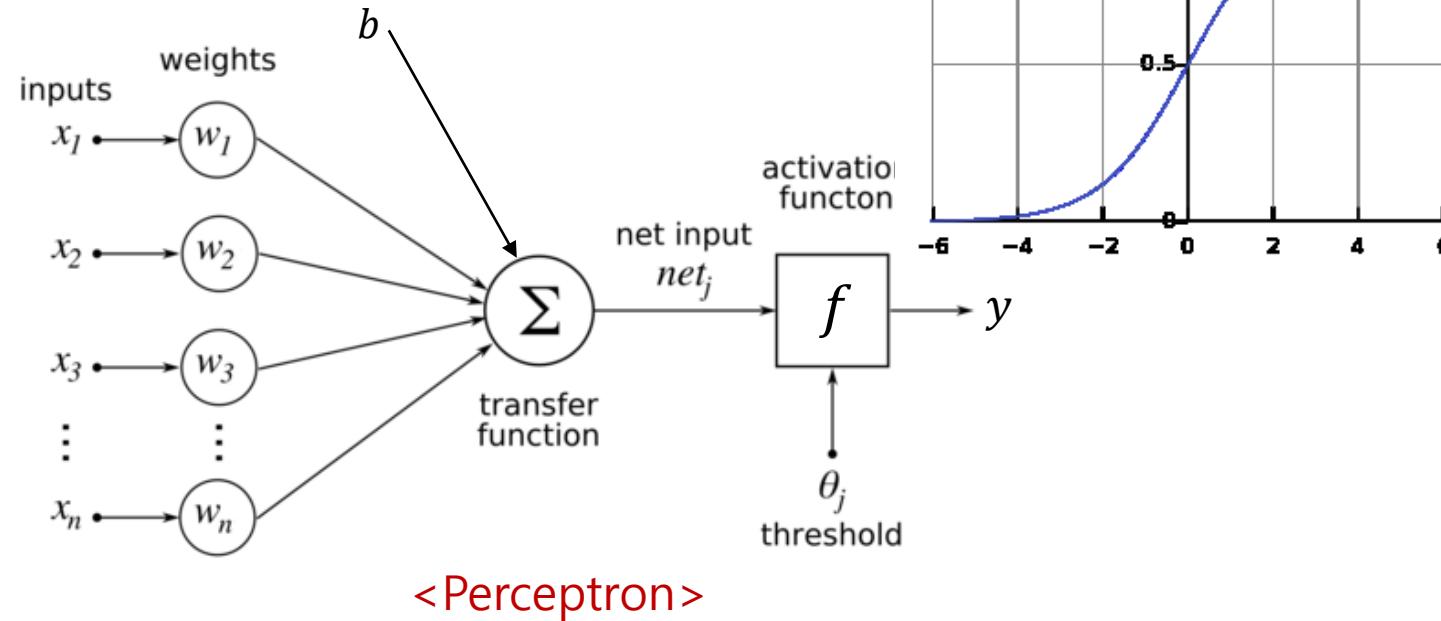


$$f(x) = \frac{1}{1 + e^{-x}}$$



<Perceptron>

# Recap – Logistic Regression



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

sigmoid  
activation

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

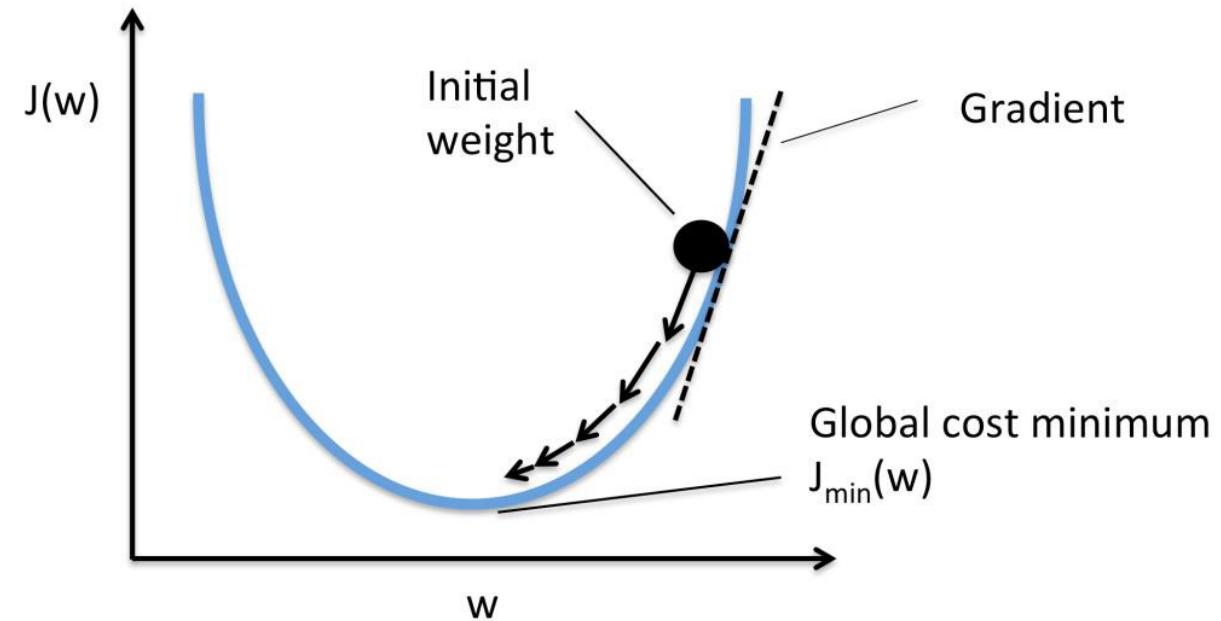
Sigmoid activation : small changes in their weights and bias cause only a small change in their output

Bias( $b$ ) : The bias is a measure of how easy it is to get the perceptron to fire

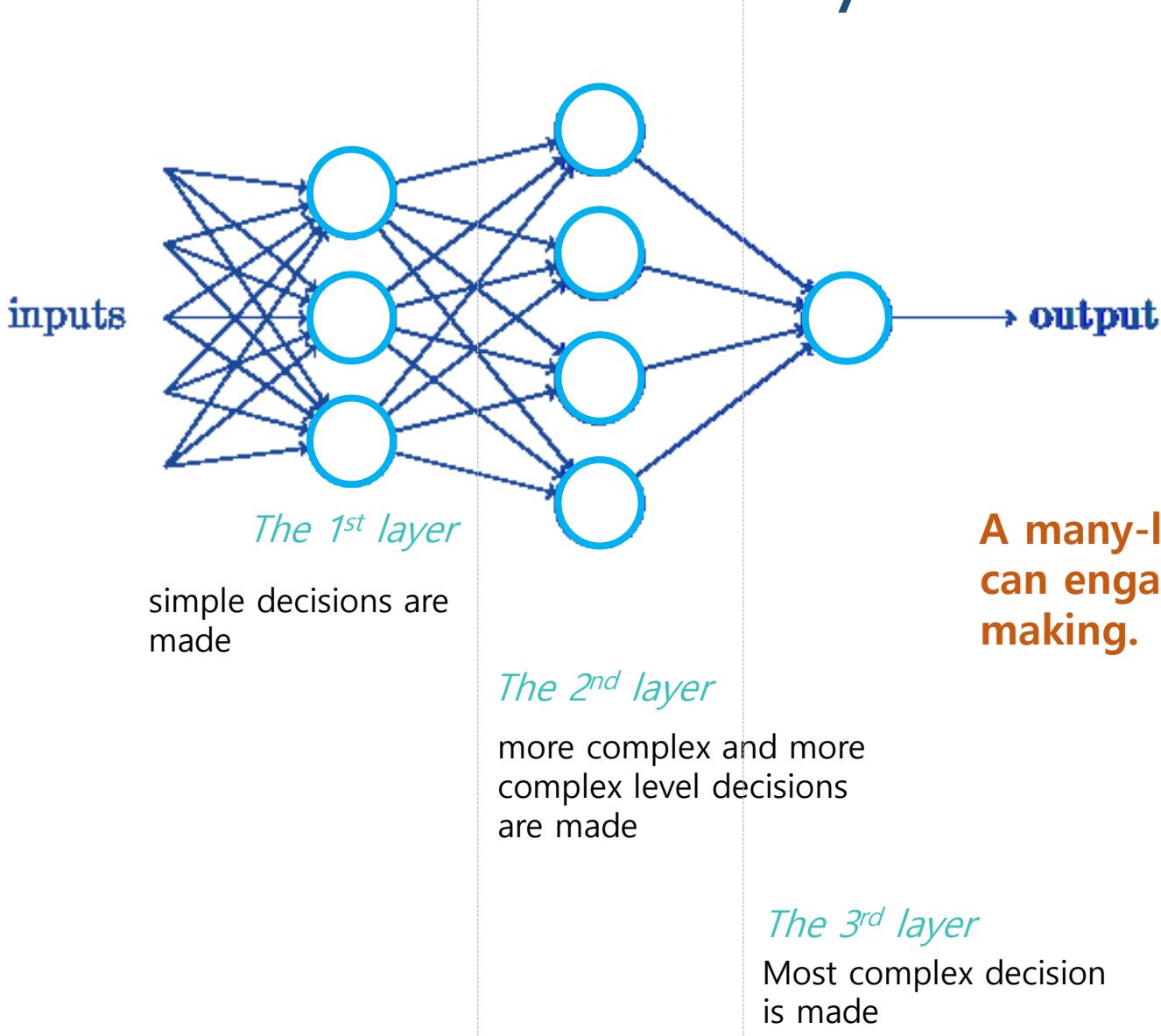
# Recap – Training(Gradient Descent)

$$w_{new} = w - \alpha \frac{\delta L}{\delta w}$$

- 방향 : 그 지점에서의 – gradient
- 속력(보폭) : learning rate( $\alpha$ )



# How about This? Multi-Layer Perceptron



Network을 **deep**하게 쌓고, **class**도 여러 개일 때는  
어떻게 학습할 수 있을까?

먼저 Multi-Layer부터 생각해봅시다

# Back Propagation!

- $w_{new} = w - \alpha \frac{\delta L}{\delta w}$ , 결국  $\frac{\delta L}{\delta w}$ 을 구하고 싶은데, input에 가까운 w로 L을 어떻게 미분할까?
- Chain Rule을 이용!

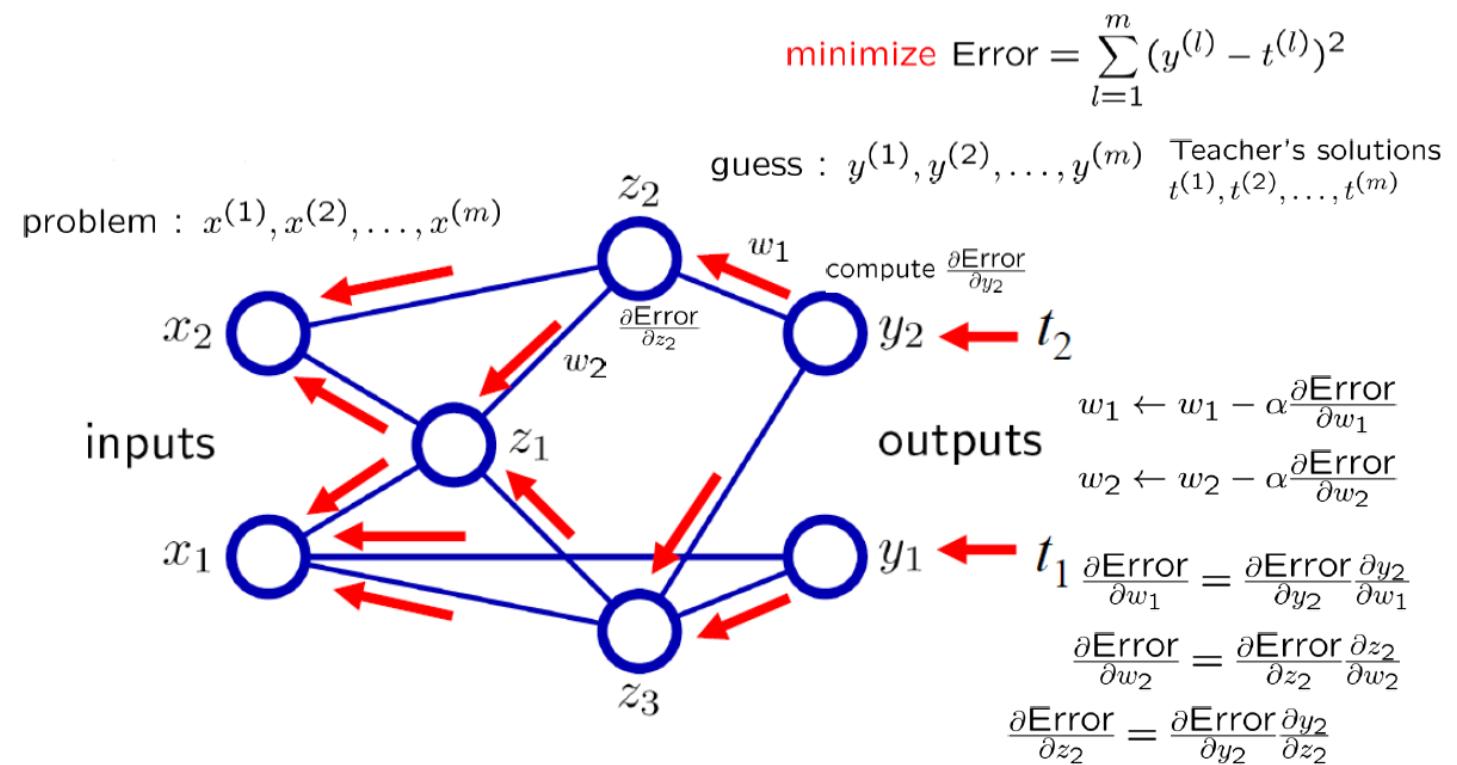
## Simple Chain Rule

$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

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

$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$

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

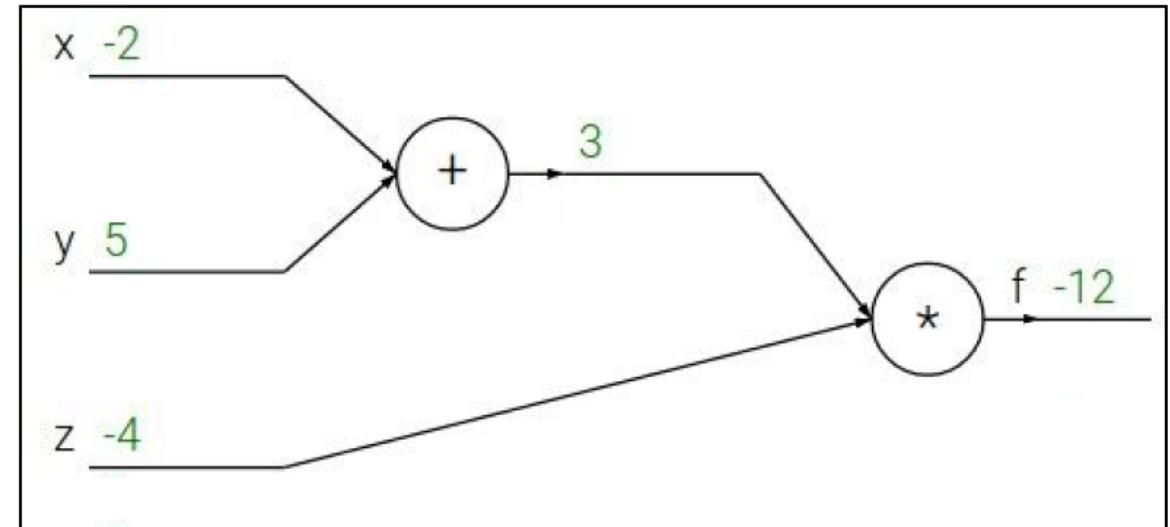


# Back Propagation

Backpropagation: a simple example

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

e.g.  $x = -2$ ,  $y = 5$ ,  $z = -4$



# Back Propagation

Backpropagation: a simple example

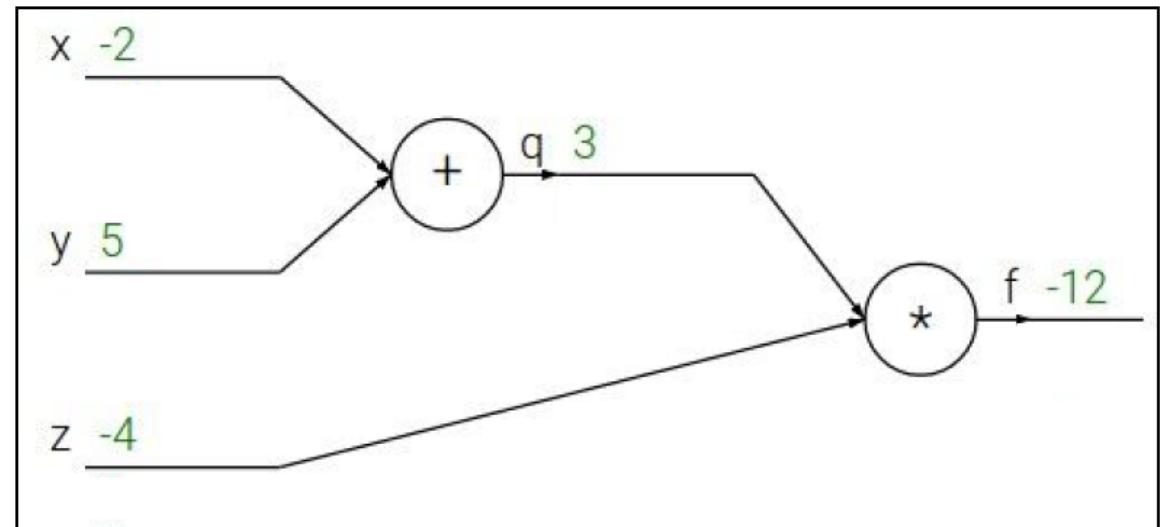
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2$ ,  $y = 5$ ,  $z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Back Propagation

Backpropagation: a simple example

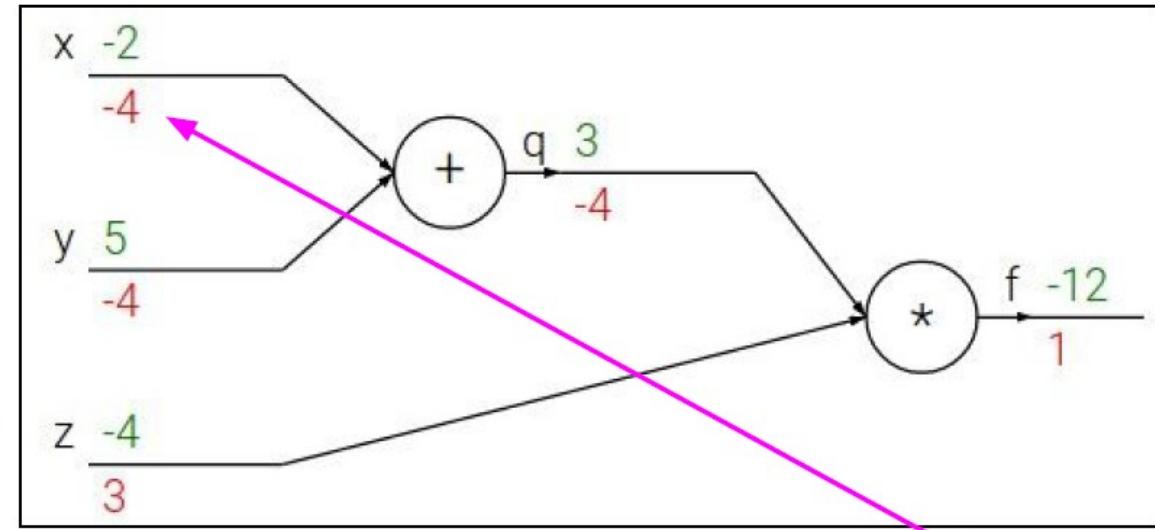
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2$ ,  $y = 5$ ,  $z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

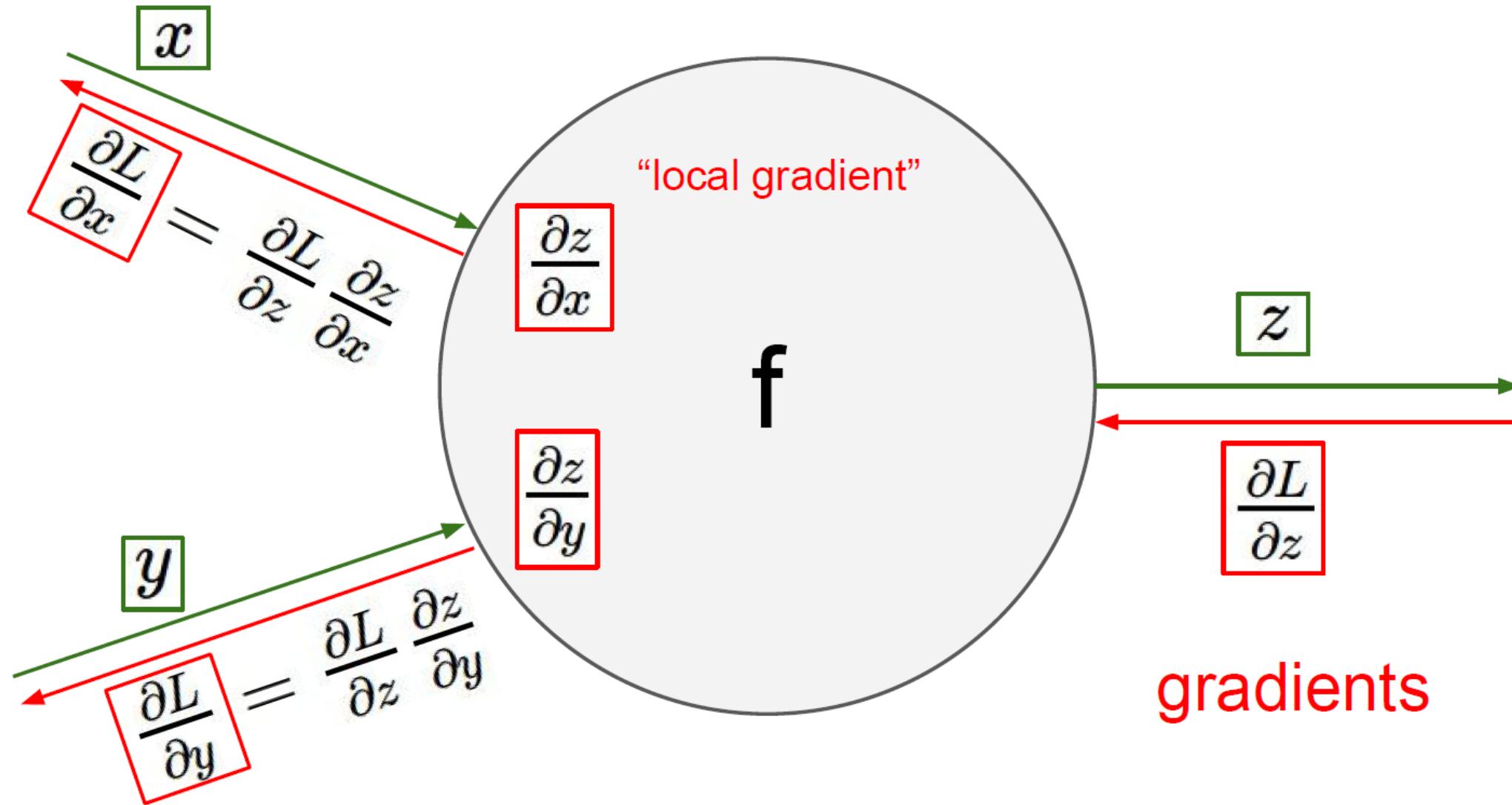


Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial x}$$

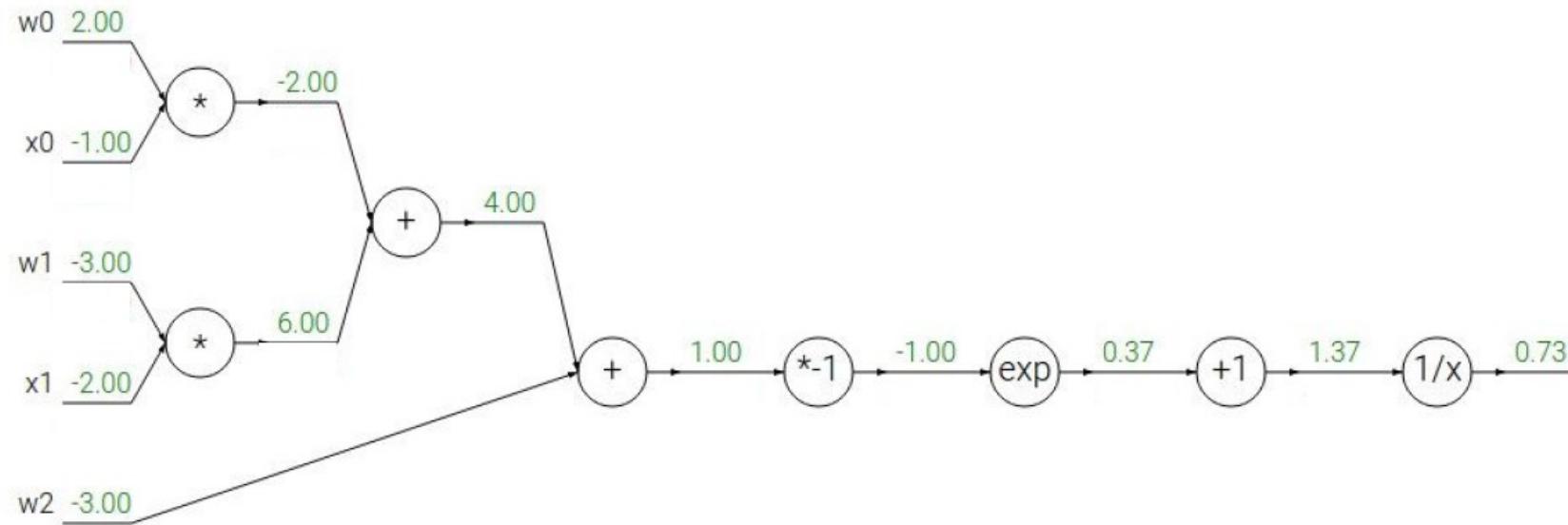
# Chain Rule(Local Gradient)



# Back Propagation(Example)

Another example:

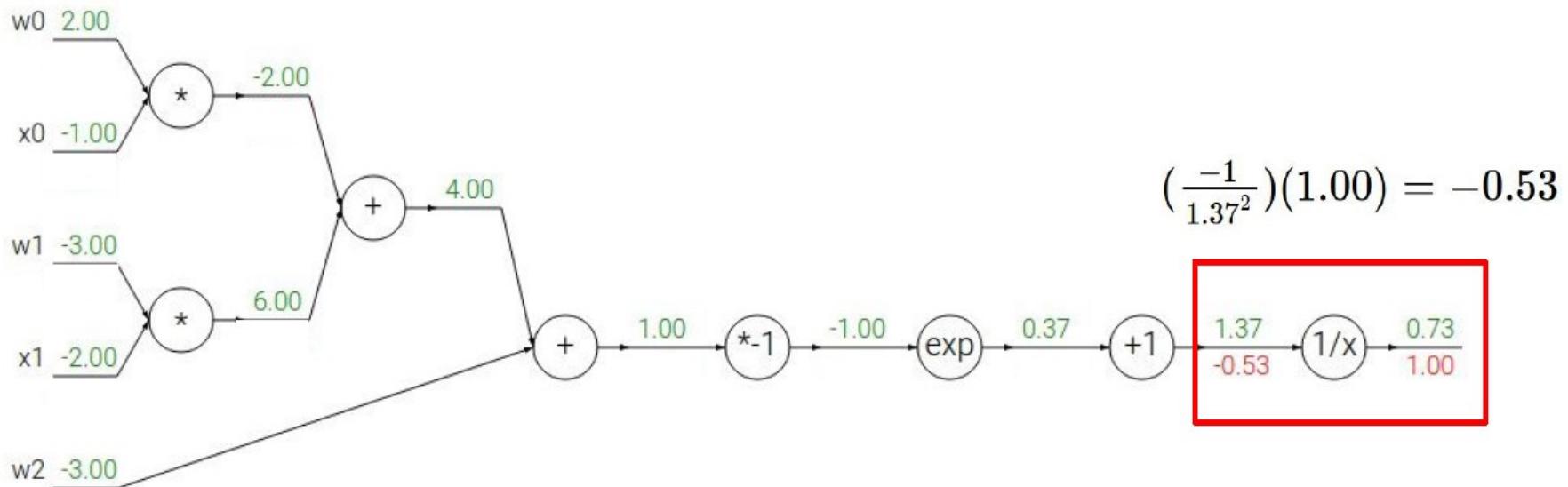
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

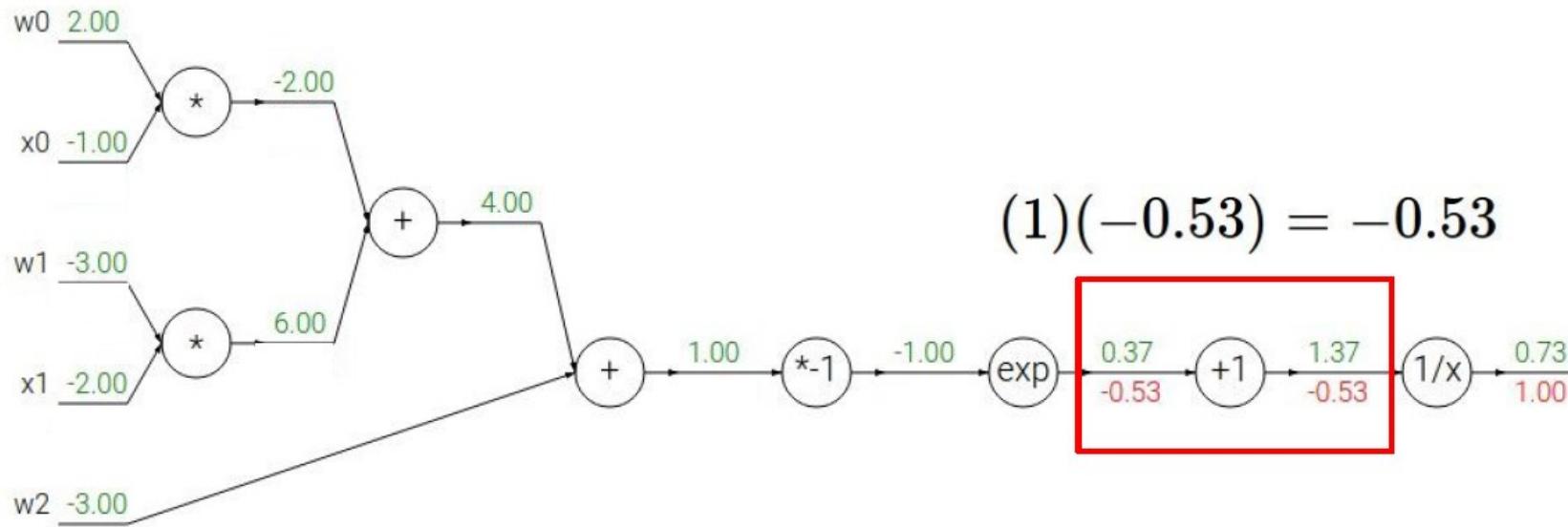
→

$$\frac{df}{dx} = 1$$

# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

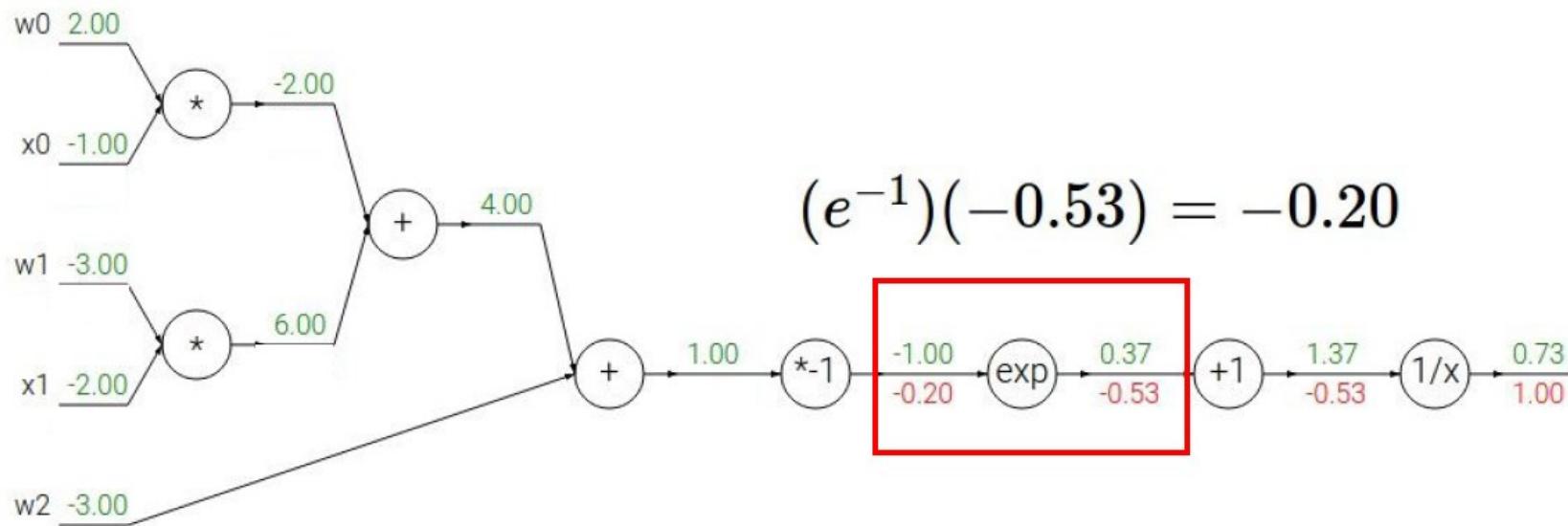
→

$$\frac{df}{dx} = 1$$

# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

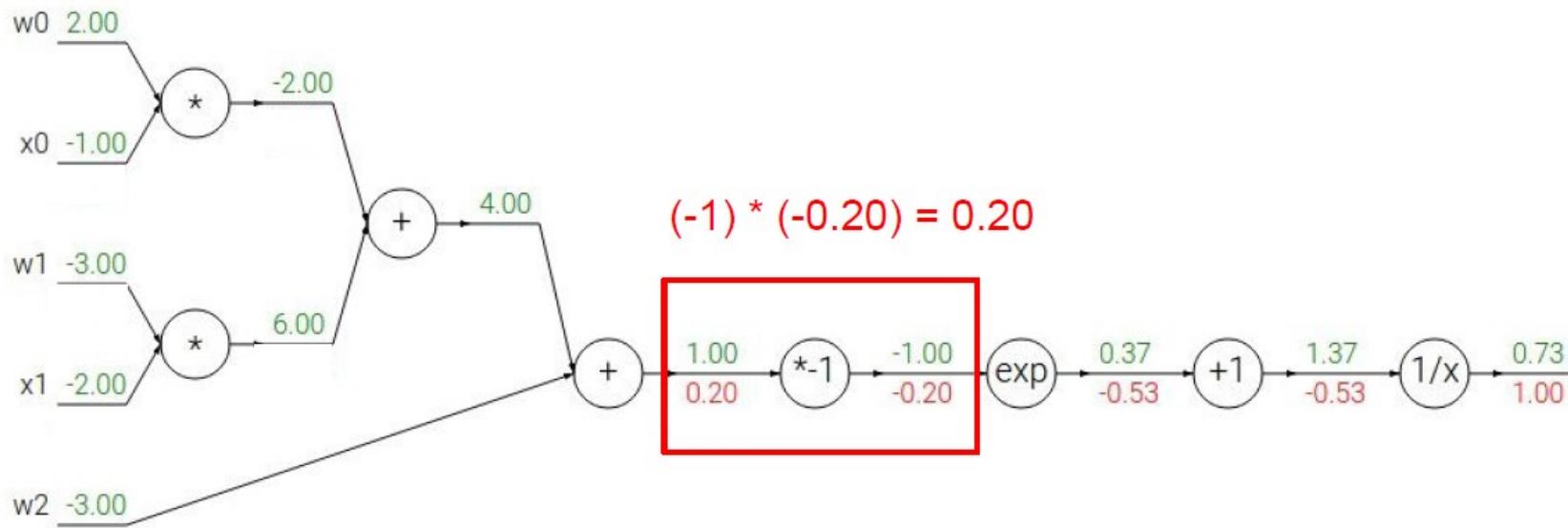
→

$$\frac{df}{dx} = 1$$

# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

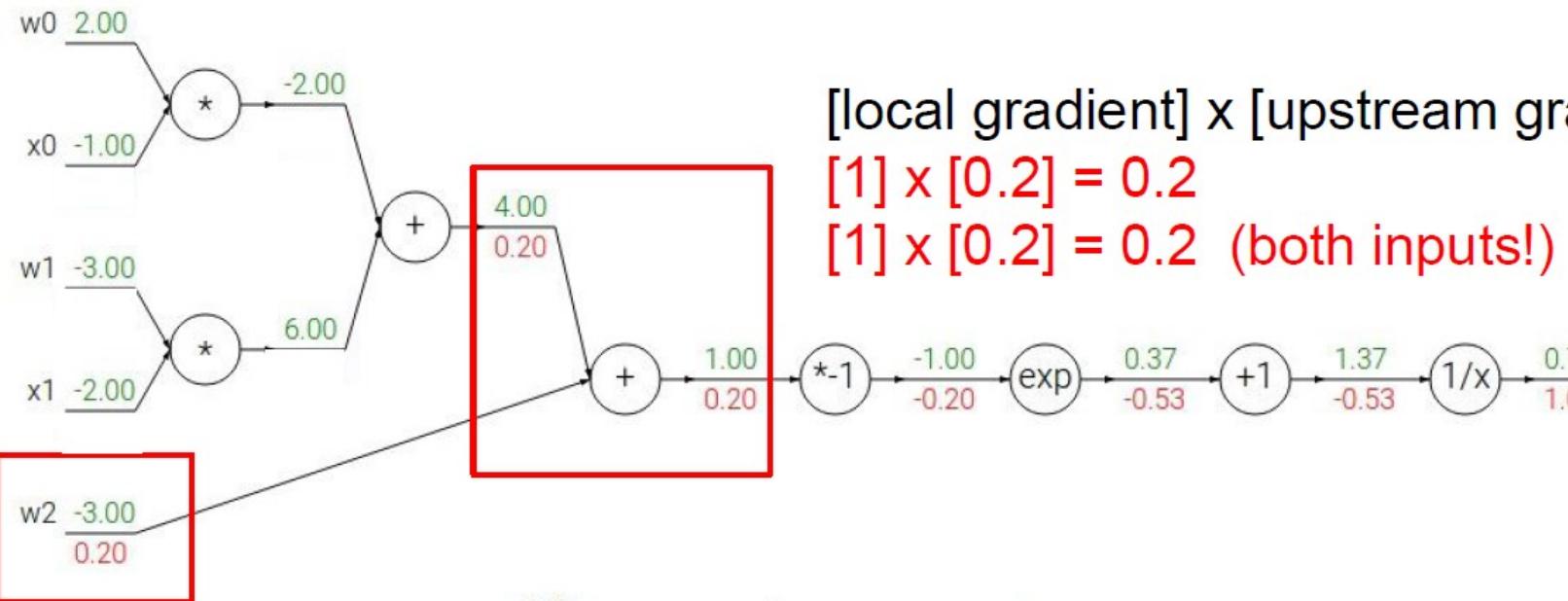
→

$$\frac{df}{dx} = 1$$

# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

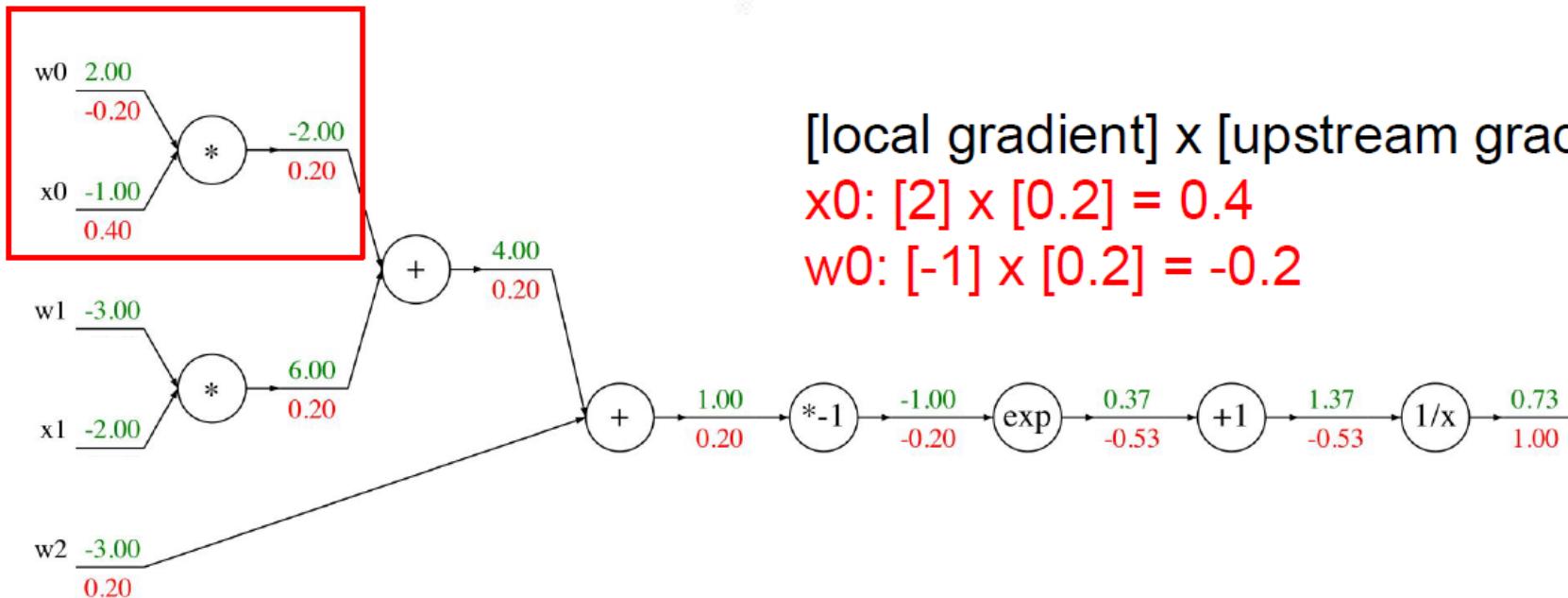
→

$$\frac{df}{dx} = 1$$

# Back Propagation(Example)

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

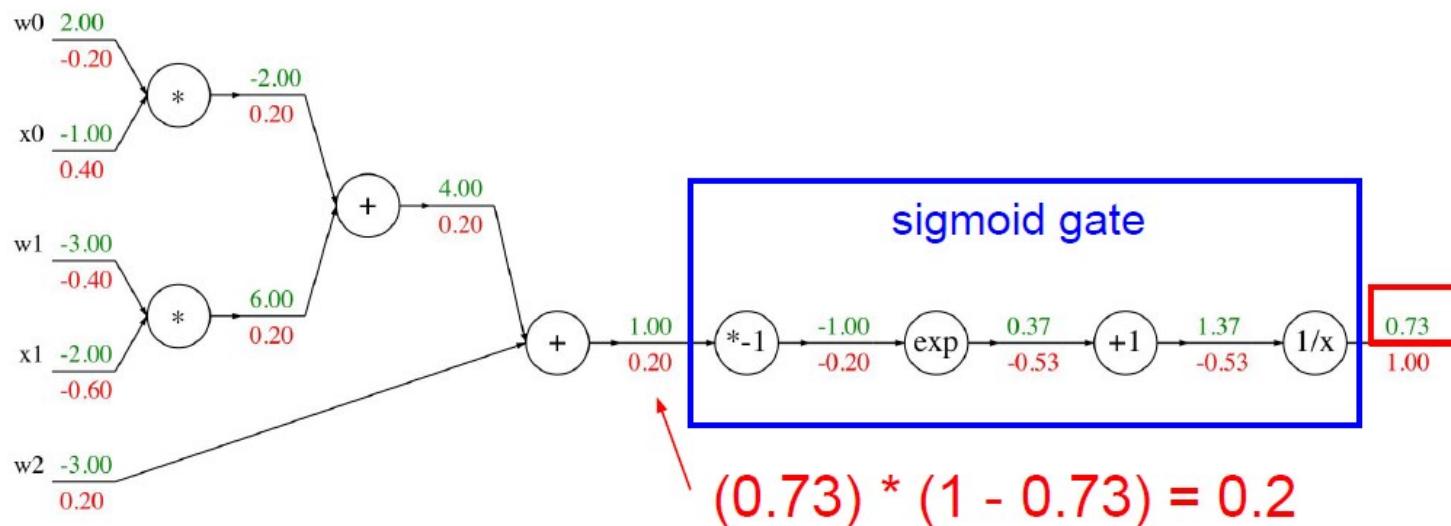
# Back Propagation(Example)

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

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

sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$



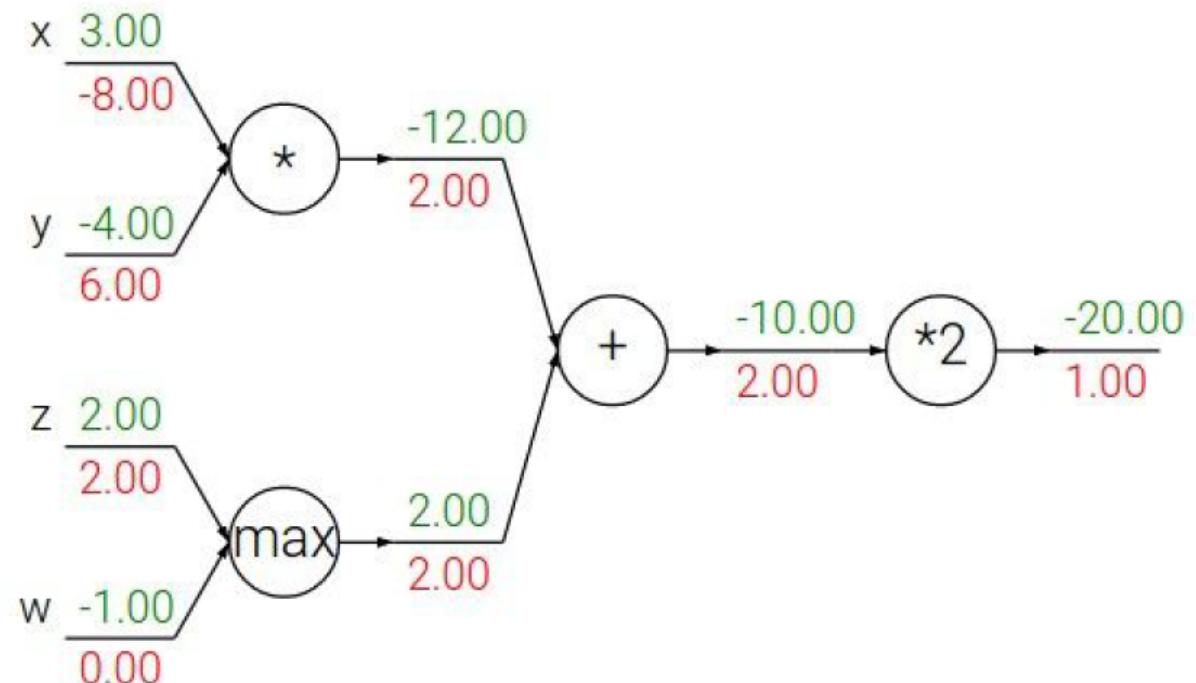
# Back Propagation(Example)

## Patterns in backward flow

**add** gate: gradient distributor

**max** gate: gradient router

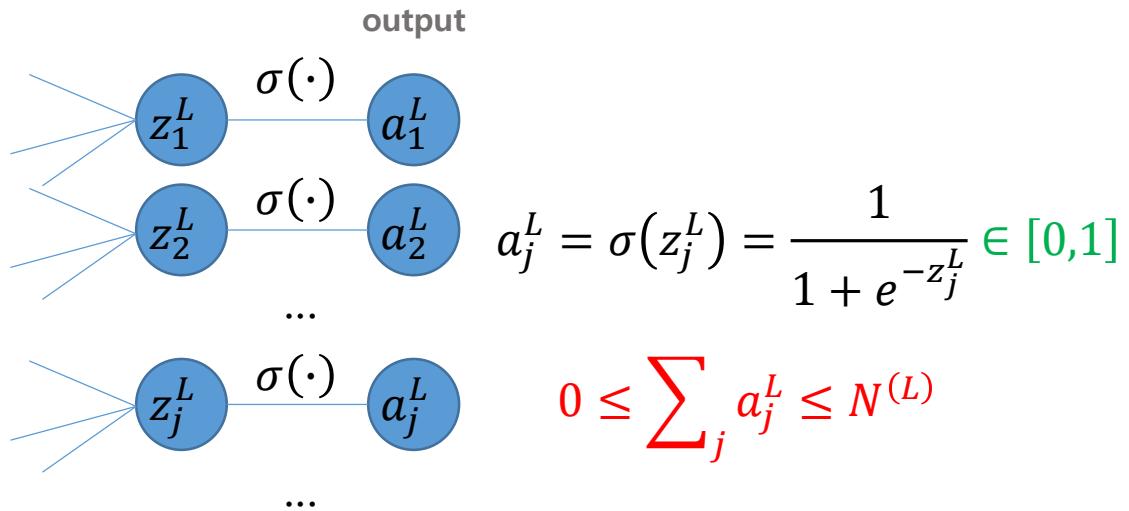
**mul** gate: gradient switcher



이제 Multi-Layer인 Network도 학습하는 방법은 알았는데, 그럼 Multi-Class는?

# Softmax!

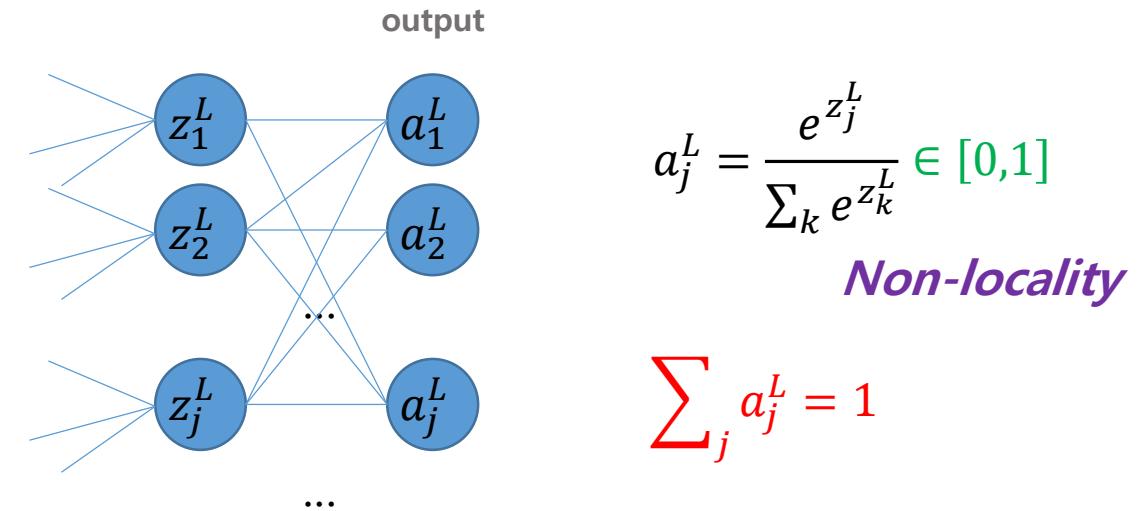
Original Output Layer



In classification problem, a desired output vector contains zero elements except only one '1' element .

→ This can be view as sum of all elements is 1

Output Layer of Softmax



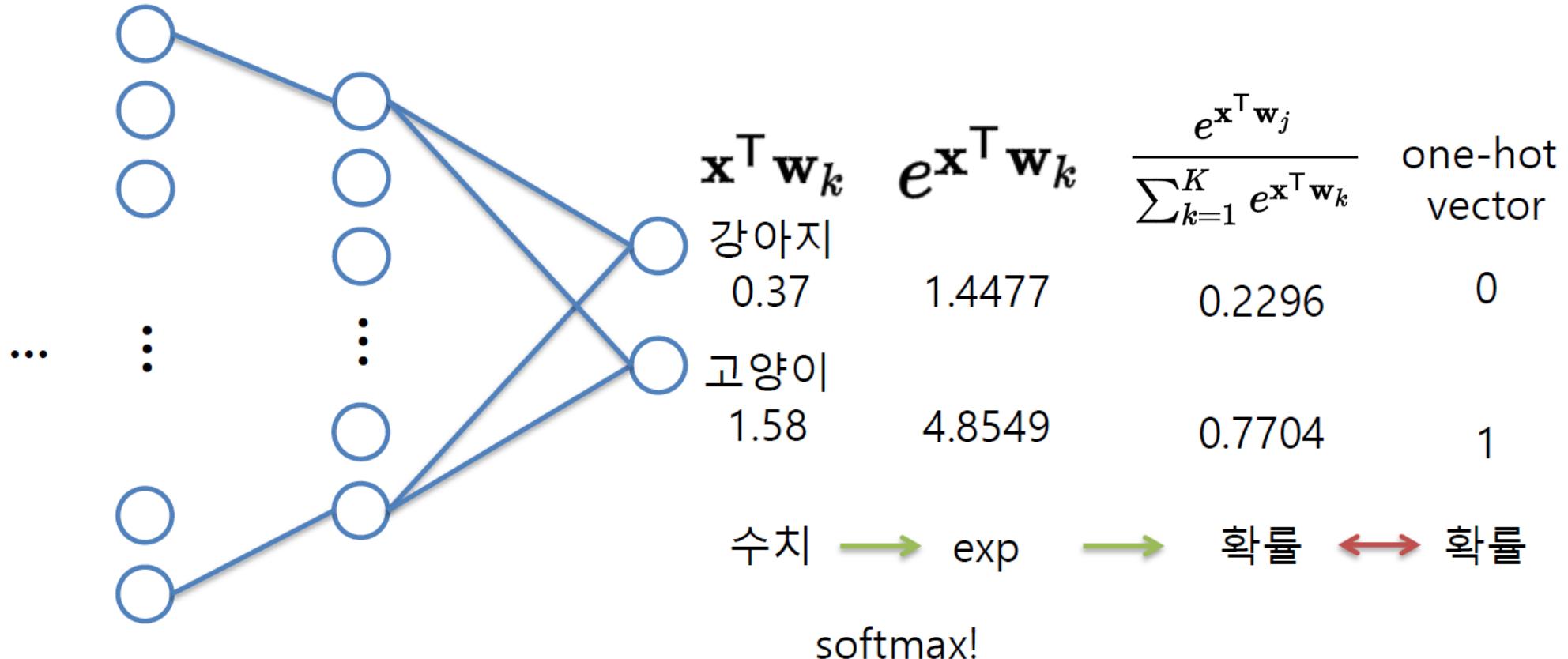
A softmax layer outputs a probability distribution!!

**Monotonicity**  $\frac{\partial a_j^L}{\partial z_k^L}$  = positive if  $j=k$ , negative otherwise

# Softmax

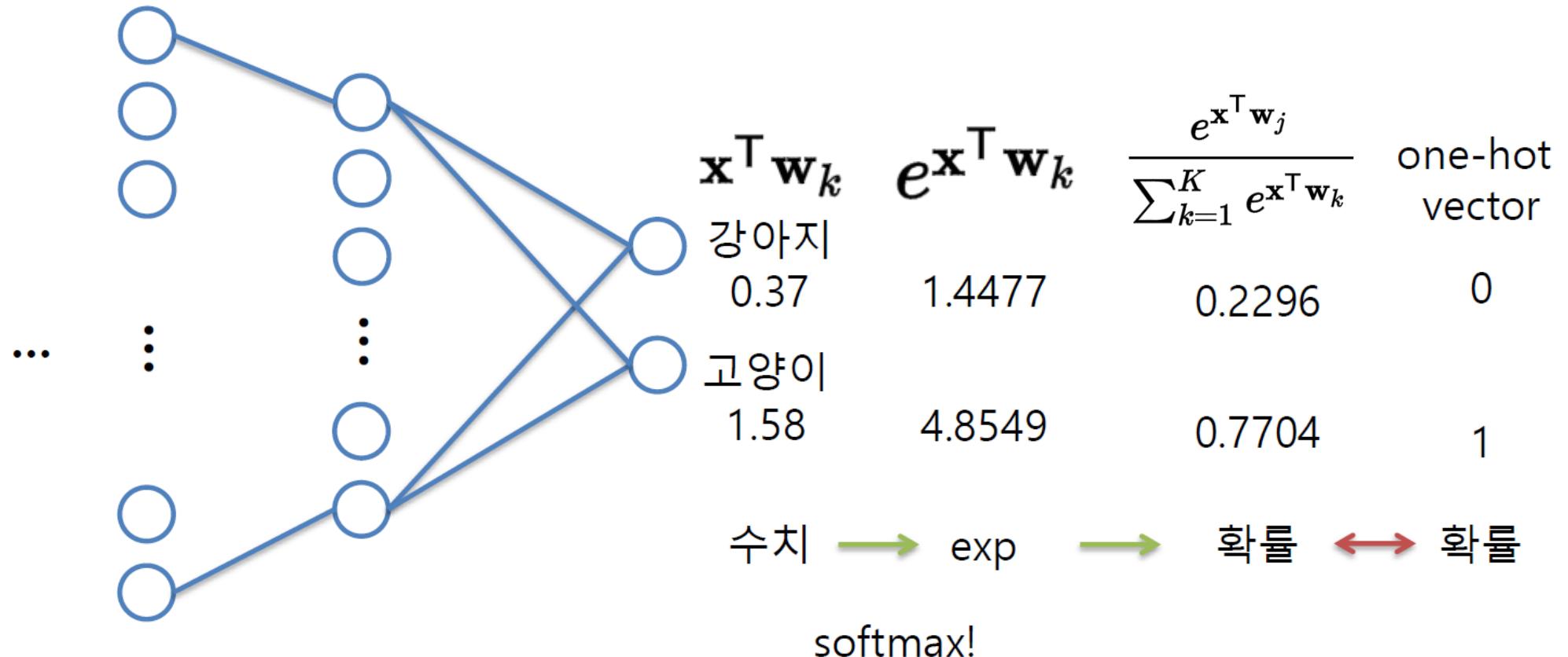
- Output을 확률처럼 나타내보자

$$P(y = j | \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}}$$



# Loss Function of Softmax

- Loss function은 어떻게 정의할까?
  - L1 loss or L2 loss(MSE)?



# Loss Function of Logistic Regression

- Cross Entropy Loss!

$$cost(W) = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$

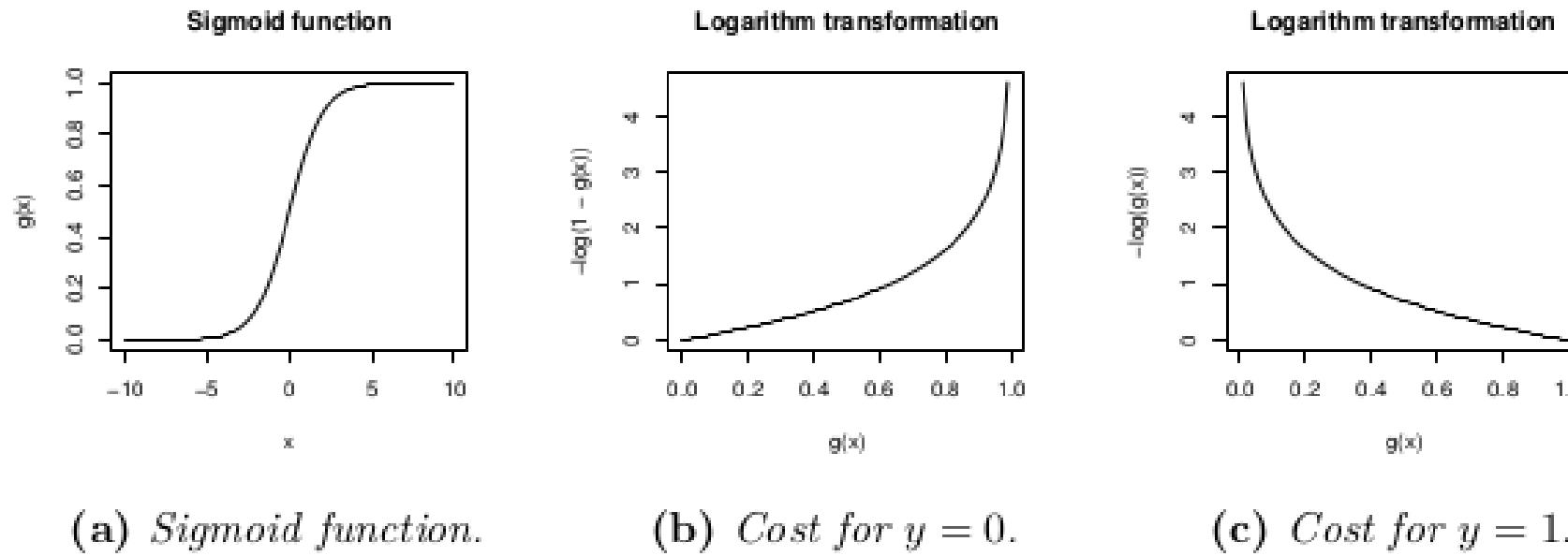
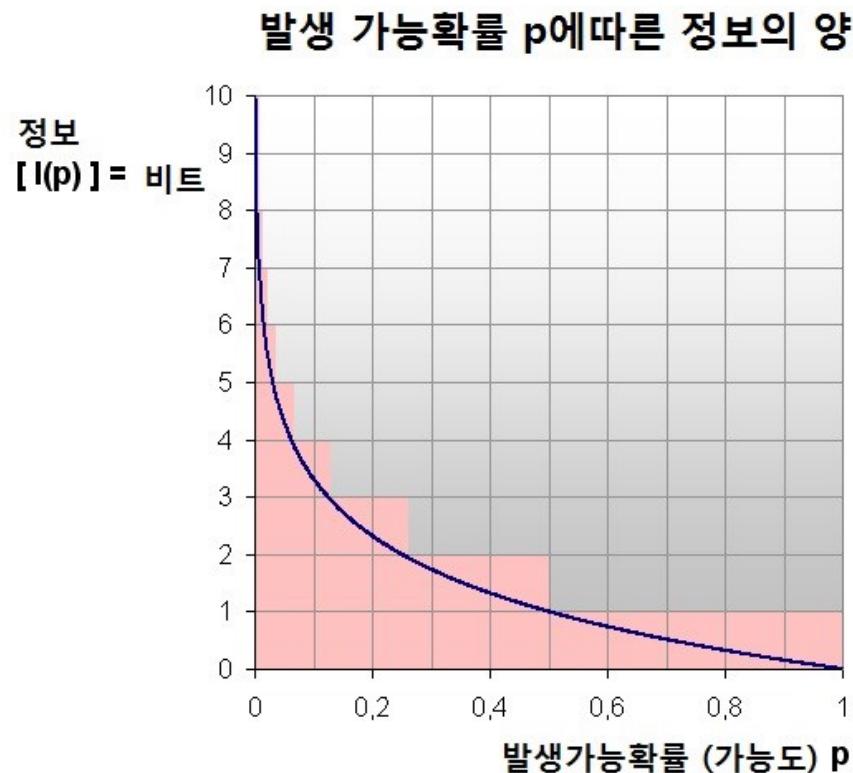


Figure B.1: Logarithmic transformation of the sigmoid function.

# entropy

- Information Theory에서 Entropy란 정보량의 기댓값(평균)이다

$$H(X) = E[I(X)] = E[-\ln(P(X))].$$

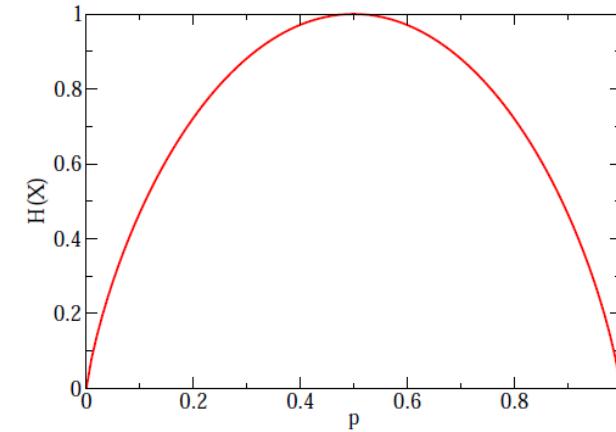


- 항상 일어나는 사건이라면  $p=1$  이고 이것이 가지고 있는 정보량은 0이다.
- 일어날 확률이 적을 수록 많은 정보를 담고 있다.

# Entropy

$$H(X) = E[I(X)] = E[-\ln(P(X))].$$

$$= \sum_{i=1}^n P(x_i) I(x_i) = - \sum_{i=1}^n P(x_i) \log_b P(x_i),$$



Ex) 동전던지기

$$P(\text{앞}) = \frac{1}{2}$$

$$P(\text{뒤}) = \frac{1}{2}$$

$$H(X) = -(0.5 * \log_2 0.5 + 0.5 * \log_2 0.5) = 0.5 + 0.5 = \mathbf{1}$$

이 확률 분포를 표현하기 위하여 평균 1 bit가 필요하다는 의미!

# Cross Entropy

True distribution p,  
Model이 예측한 distribution을 q라고 하면

$$\begin{aligned} H(p, q) &= - \sum_i p_i \log q_i \\ &= - \sum_i p_i \log q_i - \boxed{\sum_i p_i \log p_i} + \sum_i p_i \log p_i \\ &= \boxed{H(p)} + \sum_i p_i \log p_i - \sum_i p_i \log q_i \\ &= H(p) + \sum_i p_i \log \frac{p_i}{q_i} \\ &= H(p) + D_{KL}(p \parallel q) \end{aligned}$$

P 자체의 entropy(불변)  


p를 기준으로 q가 얼마나 다른가  
(p의 distribution과 q의 distribution의 거리)  


Cross entropy를 minimize 하는 것은 p와 q의 분포가 가까워지도록 만드는 것과 같다

# Cross Entropy vs MSE

	Inference(softmax output)			True Label(onehot)			Correct?
	class1	class2	class3	class1	class2	class3	
model1	0.3	0.3	0.4	0	0	1	Yes
	0.3	0.4	0.3	0	1	0	Yes
	0.1	0.2	0.7	1	0	0	No
model2	0.1	0.2	0.7	0	0	1	Yes
	0.1	0.7	0.2	0	1	0	Yes
	0.3	0.4	0.3	1	0	0	No

- 2가지 model 모두 classification error는 33%이다
- Cross entropy와 squared loss에 따른 loss 값은?

# Cross Entropy vs MSE

	Inference(softmax output)			True Label(onehot)			Correct?
	class1	class2	class3	class1	class2	class3	
model1	0.3	0.3	0.4	0	0	1	Yes
	0.3	0.4	0.3	0	1	0	Yes
	0.1	0.2	0.7	1	0	0	No
model2	0.1	0.2	0.7	0	0	1	Yes
	0.1	0.7	0.2	0	1	0	Yes
	0.3	0.4	0.3	1	0	0	No

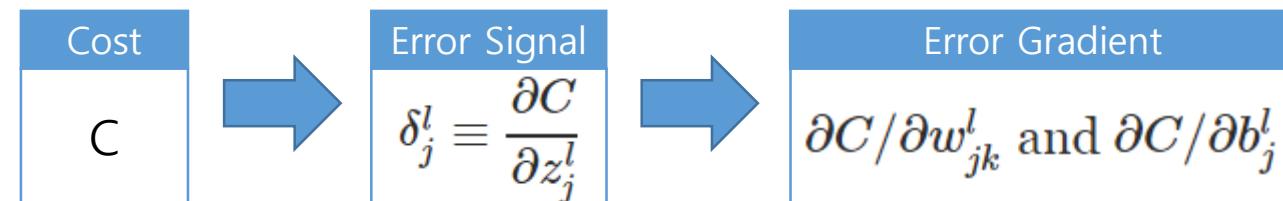
	Cross Entropy	Squared Loss
model1	$-1/3 \times (\log 0.4 + \log 0.4 + \log 0.1) = 1.38$	$1/3 \times (0.3^2 + 0.3^2 + (1-0.4)^2 + 0.3^2 + (1-0.4)^2 + 0.3^2 + (1-0.1)^2 + 0.2^2 + 0.7^2) = 0.81$
model2	$-1/3 \times (\log 0.7 + \log 0.7 + \log 0.4) = 0.64$	$1/3 \times (0.1^2 + 0.2^2 + (1-0.7)^2 + 0.1^2 + (1-0.7)^2 + 0.2^2 + (1-0.3)^2 + 0.4^2 + 0.3^2) = 0.34$

# Cross Entropy vs MSE

	Inference(softmax output)			True Label(onehot)			Correct?
	class1	class2	class3	class1	class2	class3	
model1	0.3	0.3	0.4	0	0	1	Yes
	0.3	0.4	0.3	0	1	0	Yes
	0.1	0.2	0.7	1	0	0	No
model2	0.1	0.2	0.7	0	0	1	Yes
	0.1	0.7	0.2	0	1	0	Yes
	0.3	0.4	0.3	1	0	0	No

- Cross entropy focusses on **correct classification**
- MSE focusses on **fitting values** of all classes

# Fundamental Equation behind Back-Prop

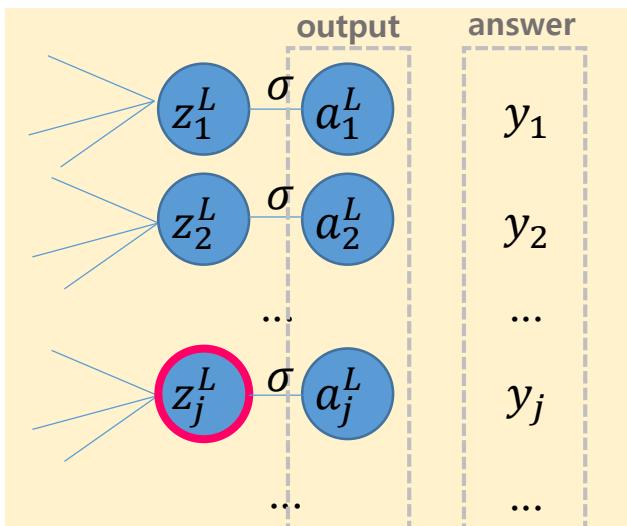


Equation for the error in the output layer,  $\delta^L$

$$\frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \cdot \sigma'(z_j^L)$$

Chain rule

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \cdot \sigma'(z_j^L)$$



$$\textbf{Quadratic Cost Function : } C = \frac{1}{2} \sum_k (y_k - a_k^L)^2$$

$$\frac{\partial C}{\partial z_j^L} = \frac{\partial}{\partial z_j^L} \left( \frac{1}{2} \sum_k (y_k - a_k^L)^2 \right) = \frac{\partial}{\partial z_j^L} \left( \frac{1}{2} \sum_k (y_k - \sigma(z_k^L))^2 \right) = \frac{\partial}{\partial z_j^L} \left( \frac{1}{2} (y_j - \sigma(z_j^L))^2 \right)$$

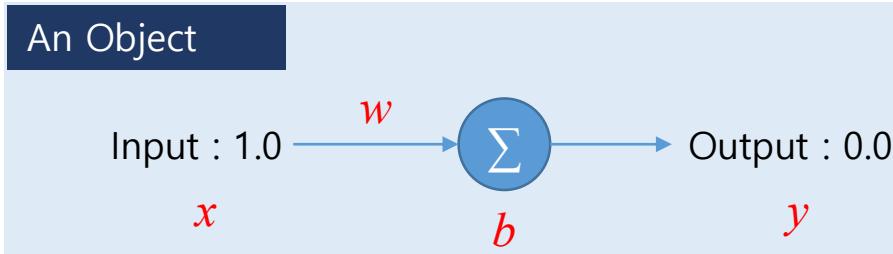
Only the  $j^{\text{th}}$  term is valid

$$\frac{\partial C}{\partial z_j^L} = -(y_j - \sigma(z_j^L)) \sigma'(z_j^L) = (a_j^L - y_j) \sigma'(z_j^L)$$

# Mean Squared Error

We hope and expect that our neural networks will learn fast from their errors.

An example :



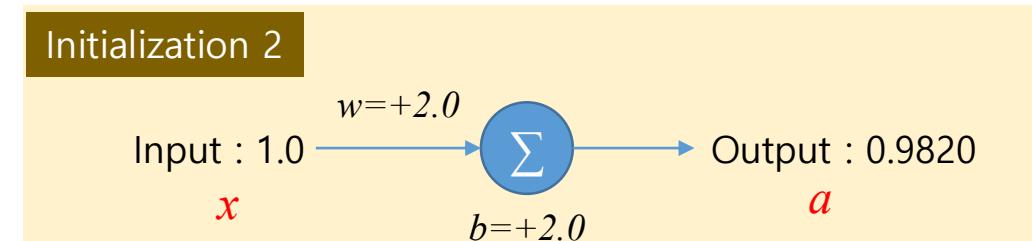
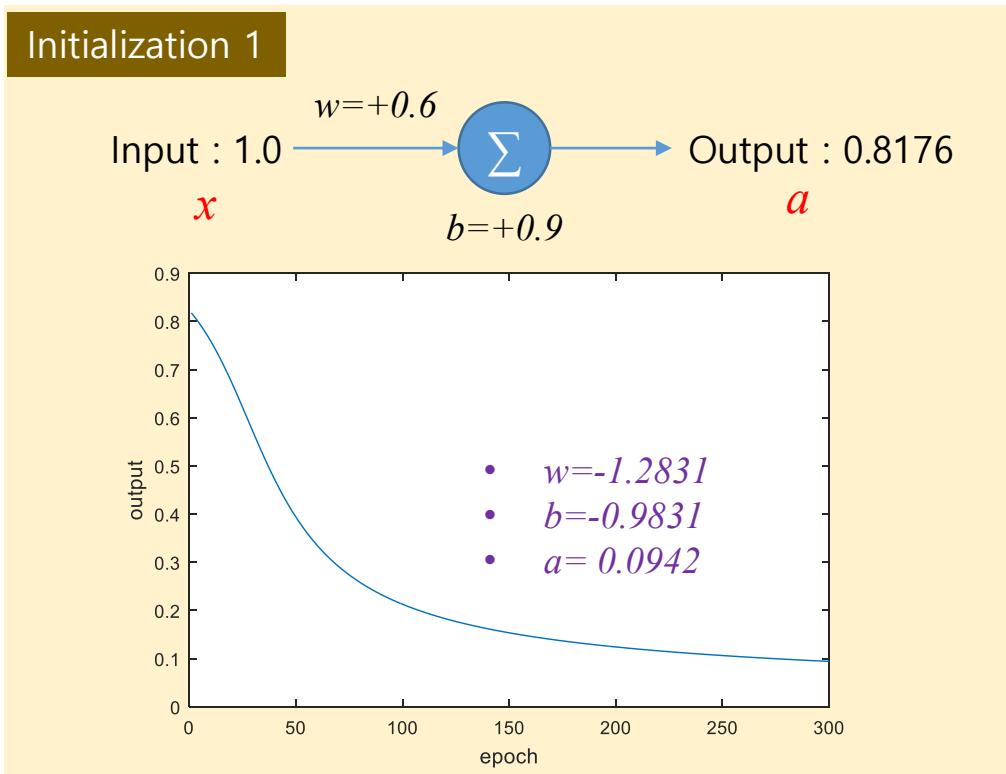
$$C = \frac{(a - y)^2}{2} = \frac{a^2}{2}$$

$$a = \sigma(z) = \sigma(wx + b) = \sigma(w + b)$$

$$\delta = a\sigma'(w + b)$$

$$w = w - \eta\delta$$

$$b = b - \eta\delta$$

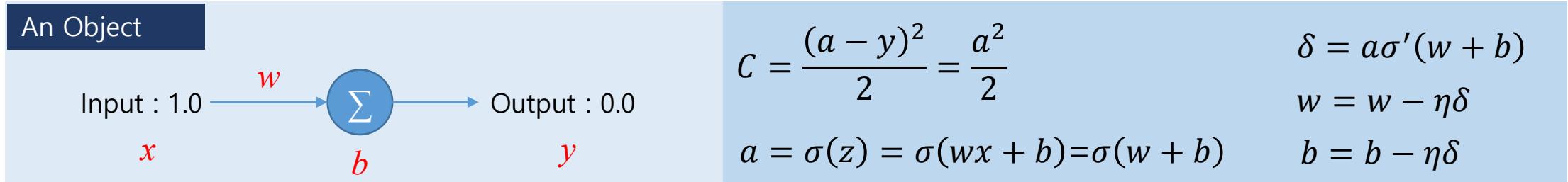


difficulty learning when it's badly wrong

# Mean Squared Error

We hope and expect that our neural networks will learn fast from their errors.

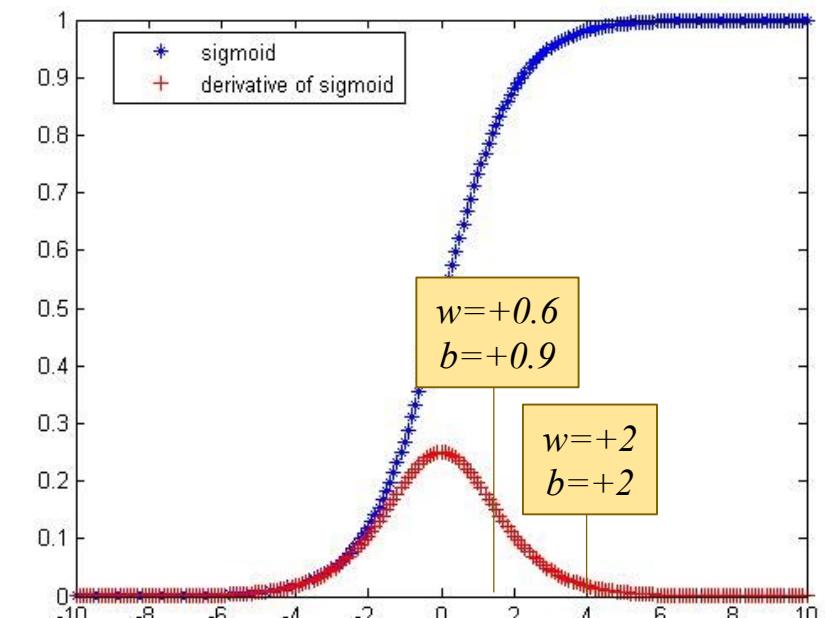
An example :



Learning slow means are  $\partial C / \partial w$ ,  $\partial C / \partial b$  small

Why they are small???

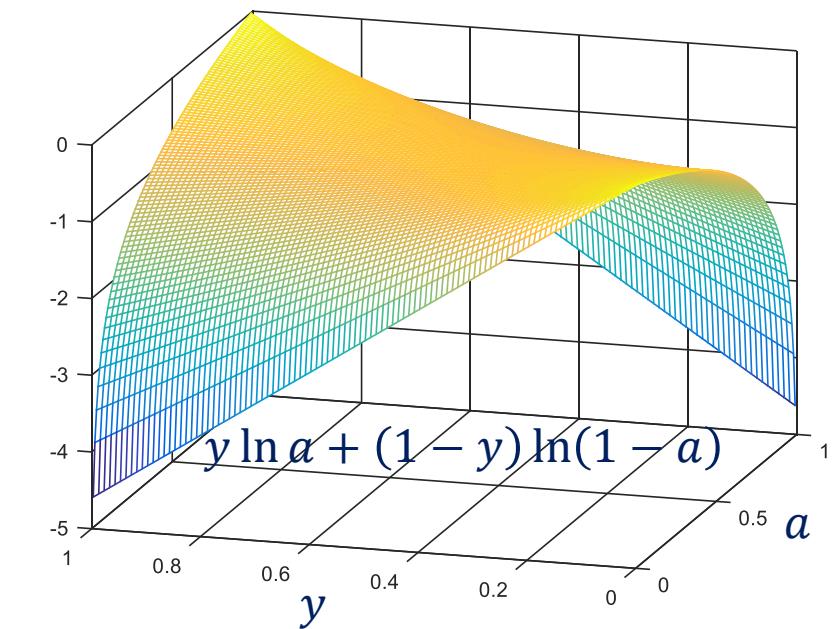
$$C = \frac{(y - a)^2}{2}$$
$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) = a\sigma'(z)$$



# Cross Entropy Cost Function

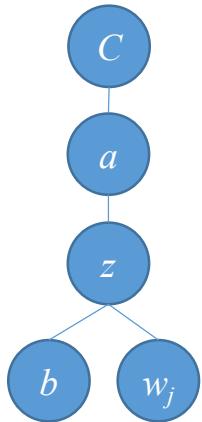
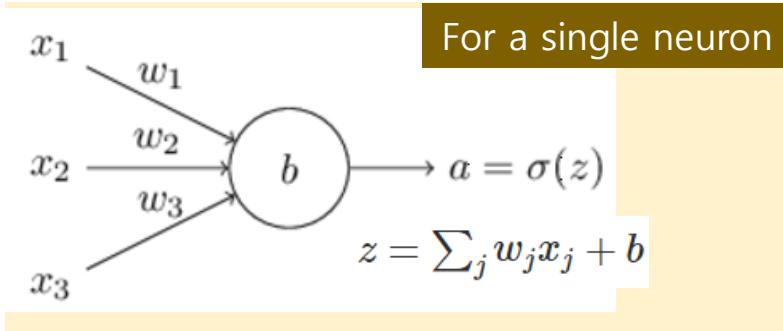
Introducing the cross-entropy cost function

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



[Two features]

- Non-negative → cost function
- Zero at  $y=a$



Error gradient

$$\frac{\partial C}{\partial w_j} = \frac{-1}{n} \sum_x \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

$$\frac{\partial C}{\partial a} = \frac{y}{a} + (1 - y) \frac{-1}{1 - a} = \frac{y - a}{(1 - a)a}$$

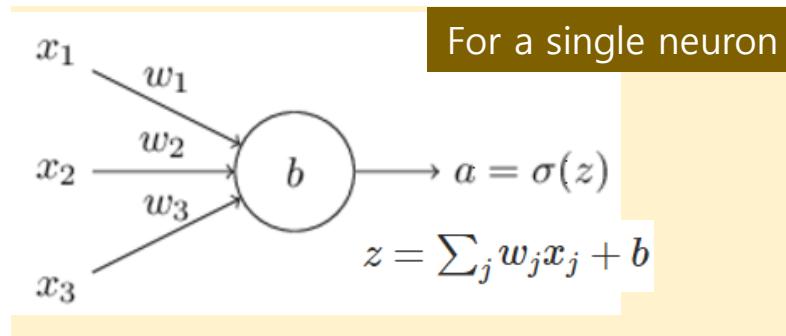
$$\frac{\partial a}{\partial z} = \sigma'(z) = (1 - \sigma(z))\sigma(z) = (1 - a)a \quad \frac{\partial z}{\partial w_j} = x_j$$

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

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

# Cross Entropy vs MSE

## Error gradient comparison



### Cross – entropy cost

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

### Quadratic cost

$$C = \frac{1}{n} \sum_x \frac{1}{2} (a - y)^2$$

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

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

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

Error signal dereaser!!!

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y) \cdot \boxed{\sigma'(z)}$$

# Appendix: Softmax Function (K-class classification)

when we consider K-class classification, posterior function can be derive as below

$$\begin{aligned} P(Y_1|X) &= \frac{\exp(a_1)}{\exp(a_1) + \exp(a_2) + \dots + \exp(a_k)} \\ &= \frac{\exp(a_1)}{\sum_{j=1}^K \exp(a_j)} \end{aligned}$$

$$a_k = \ln(P(x|c_k)P(c_k)), k = 1, 2$$

$$P(c_1|x) = \frac{\exp(a_1)}{\exp(a_1) + \exp(a_2)}$$

: 2-class classification

→ : softmax function

$$P(Y_k|X) = \frac{\exp(-\theta_k x)}{\sum_{j=1}^K \exp(-\theta_j x)}$$