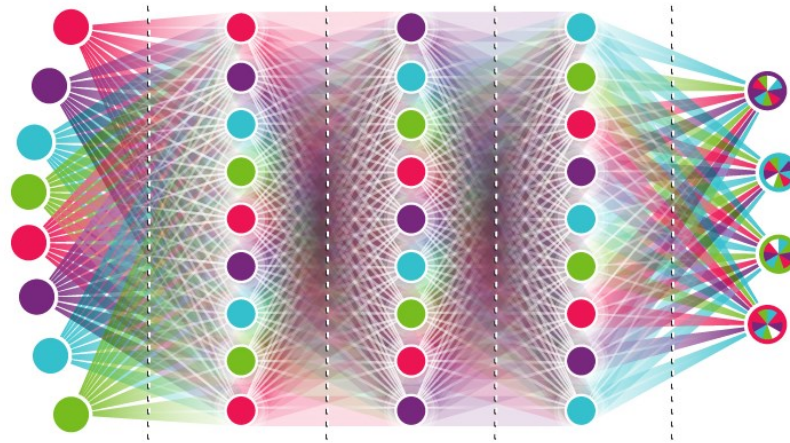
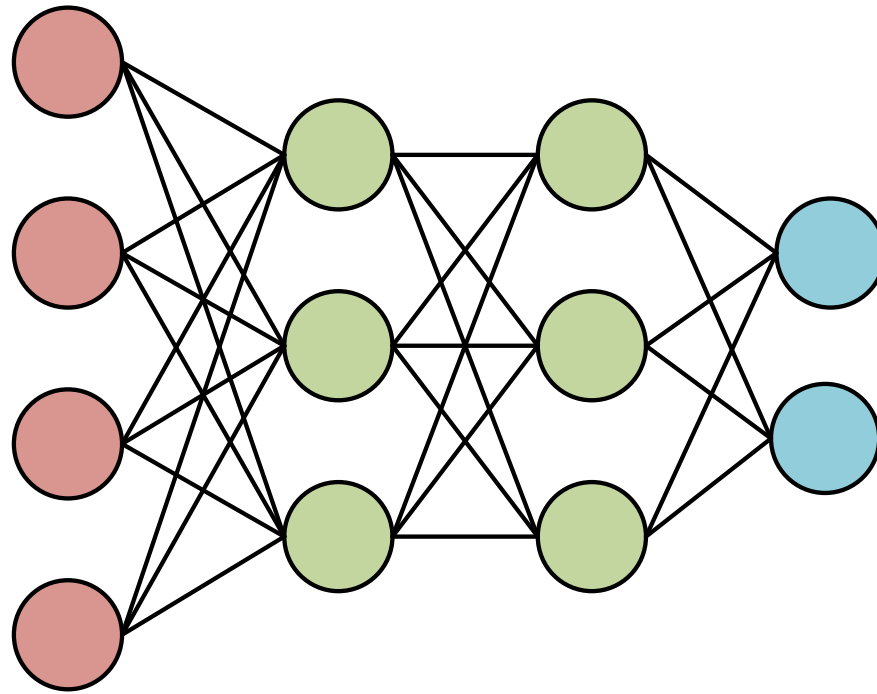


Multi-Layer Perceptron



Multi-Layer Perceptron



A many-layer network of perceptrons can engage in sophisticated decision making.

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

먼저 **Multi-Layer**부터 생각해봅시다

미분을 계산해봅시다!

$$z_{11} = x_1 \cdot w_{11} + x_2 \cdot w_{12} + x_3 \cdot w_{13} + x_4 \cdot w_{14}$$

$$a_{11} = \sigma(z_{11}) = \frac{1}{1 + e^{-z_{11}}}$$

$$z_2 = a_{11} \cdot w_{21} + a_{12} \cdot w_{22} + a_{13} \cdot w_{23}$$

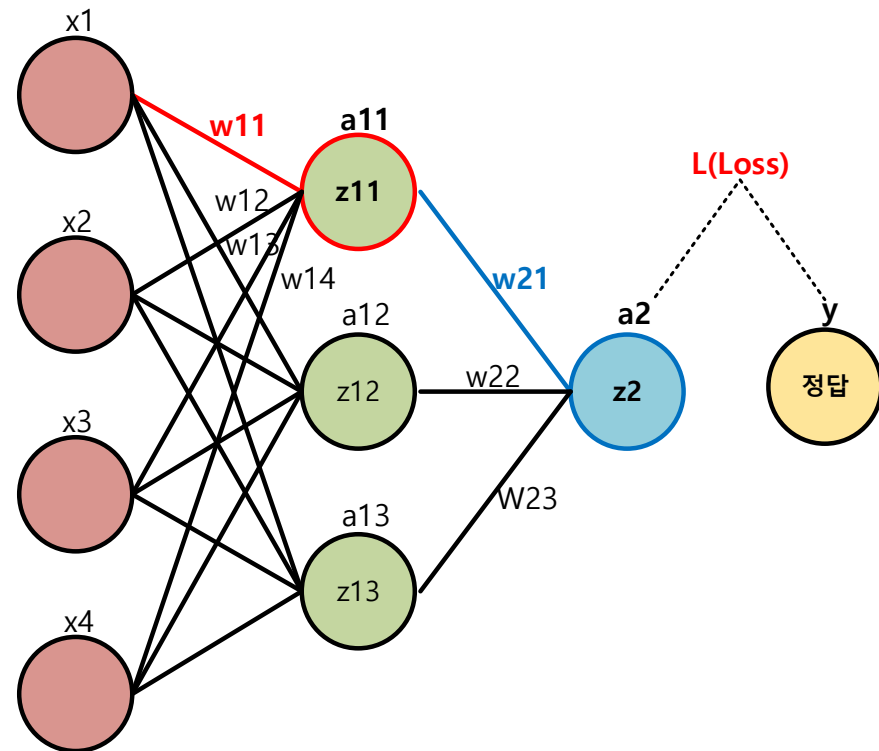
$$a_2 = z_2$$

$$L = (y - a_2)^2$$

일 때,

w11을 update 하기 위해 필요한 미분값

$$\frac{\partial L}{\partial w_{11}} = \text{??????}$$



Back Propagation

$$\begin{aligned}z_{11} &= x_1 \cdot w_{11} + x_2 \cdot w_{12} + x_3 \cdot w_{13} + x_4 \cdot w_{14} \\a_{11} &= \sigma(z_{11}) = \frac{1}{1 + e^{-z_{11}}} \\z_2 &= a_{11} \cdot w_{21} + a_{12} \cdot w_{22} + a_{13} \cdot w_{23} \\a_2 &= z_2 \\L &= (y - a_2)^2\end{aligned}$$

Loss부터 거꾸로 한 단계씩 미분을 해봅시다

$$\partial L / \partial a_2 = -2(y - a_2)$$

$$\partial a_2 / \partial z_2 = 1$$

$$\partial z_2 / \partial a_{11} = w_{21}$$

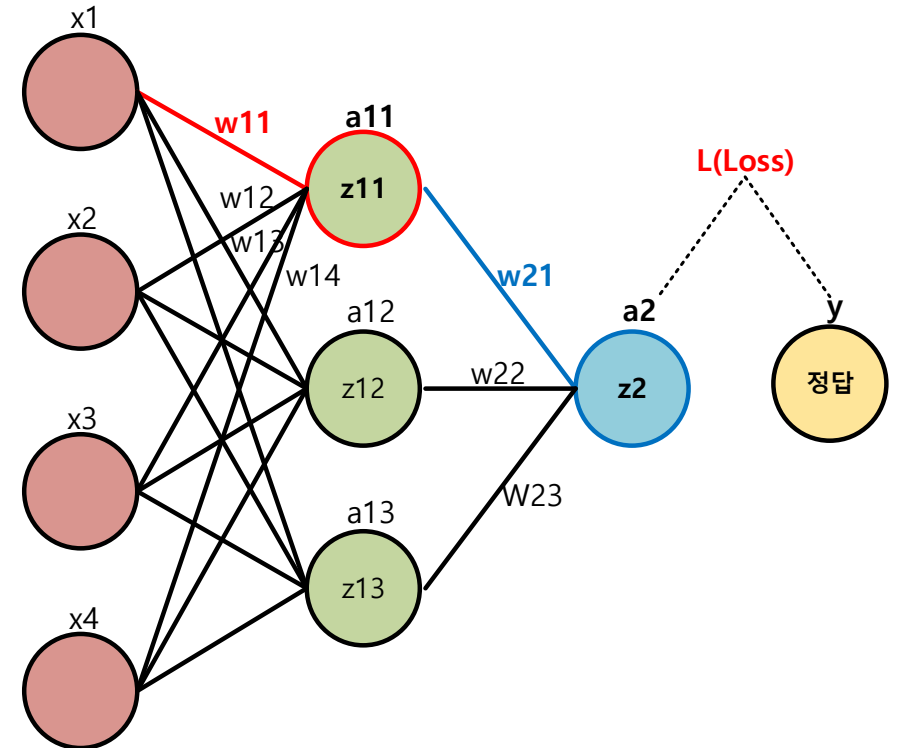
$$\partial a_{11} / \partial z_{11} = \sigma(z_{11}) \cdot (1 - \sigma(z_{11}))$$

$$\partial z_{11} / \partial w_{11} = x_1$$

이 미분들을 전부 각각 곱하면(chain rule),

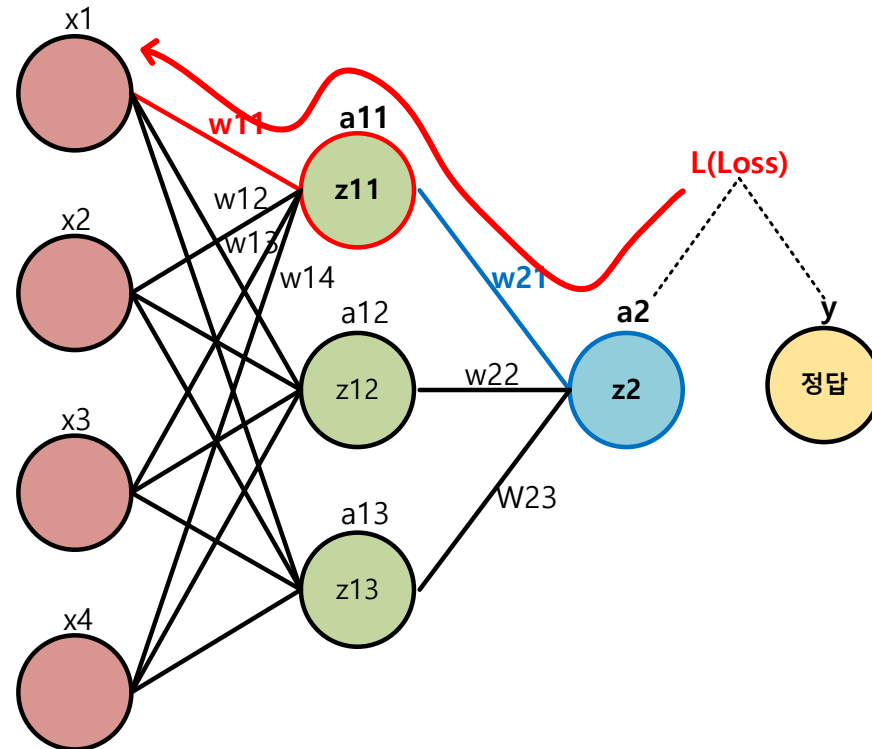
$$\frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{11}} = \frac{\partial L}{\partial w_{11}}$$

우리가 구하려고 했던 미분값



Back Propagation

Loss로부터 거꾸로 한 단계씩 미분 값을 구하고 이 값들을 chain rule에 의하여 곱해가면서 weight에 대한 gradient를 구하는 방법

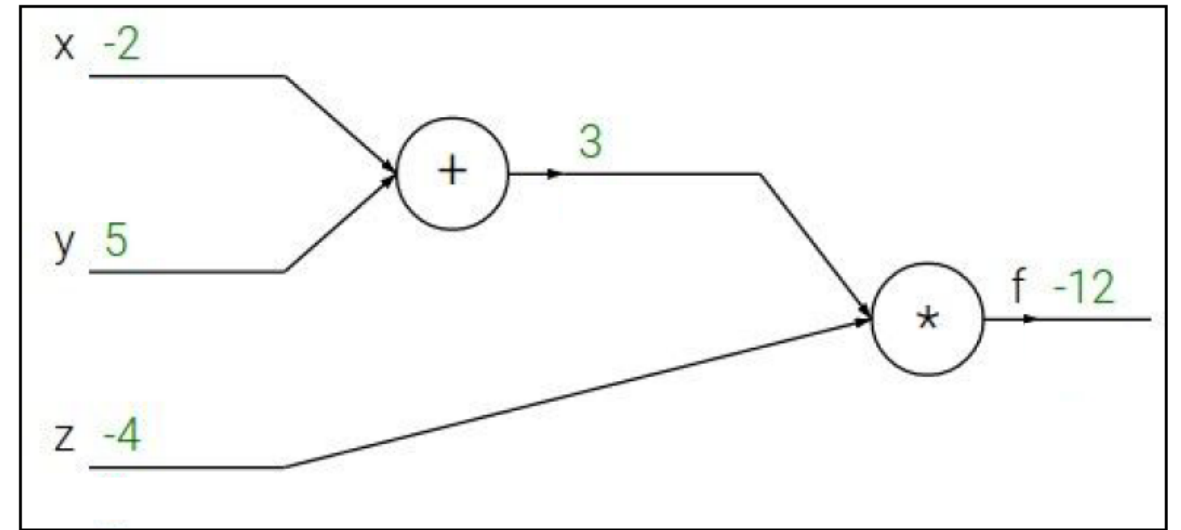


Back Propagation

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

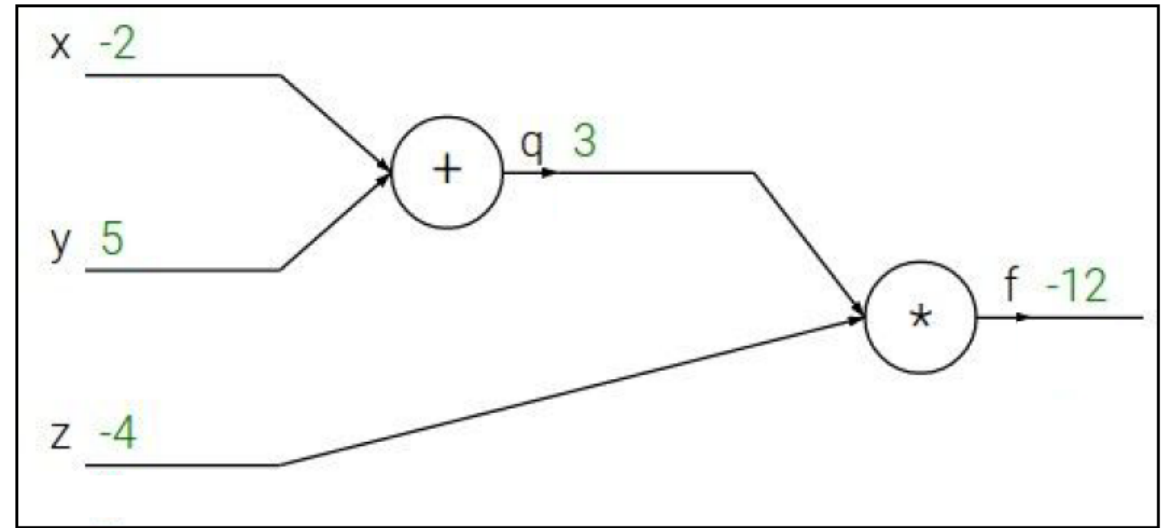
$$f(x, y, z) = (x + y)z$$

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

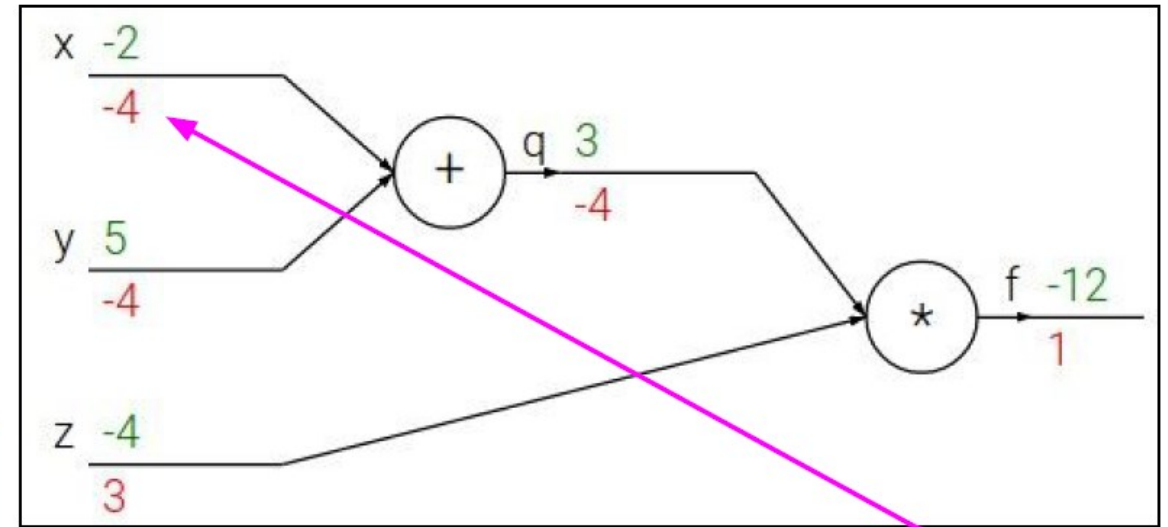
$$f(x, y, z) = (x + y)z$$

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

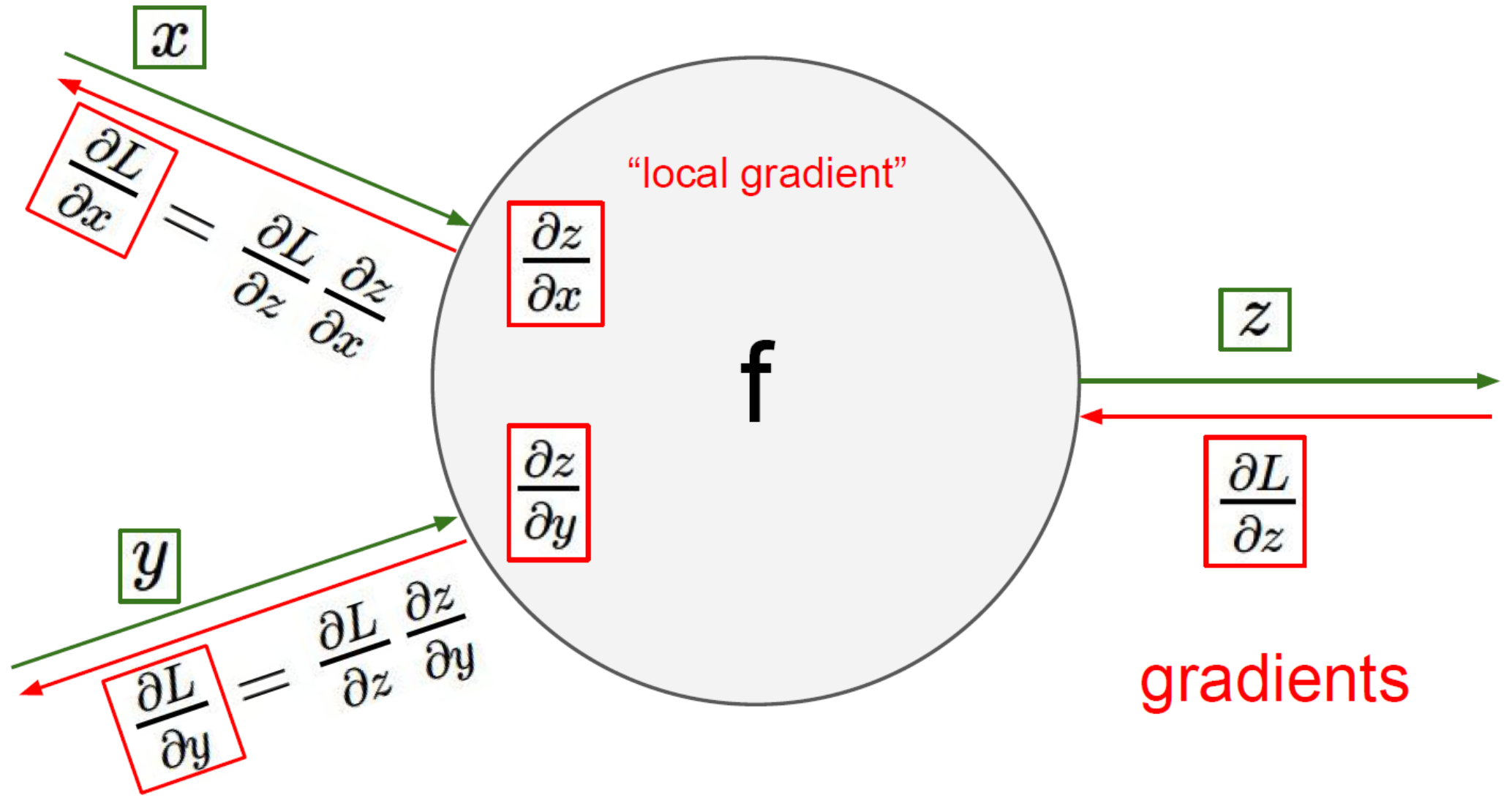


$$\frac{\partial f}{\partial x}$$

Chain rule:

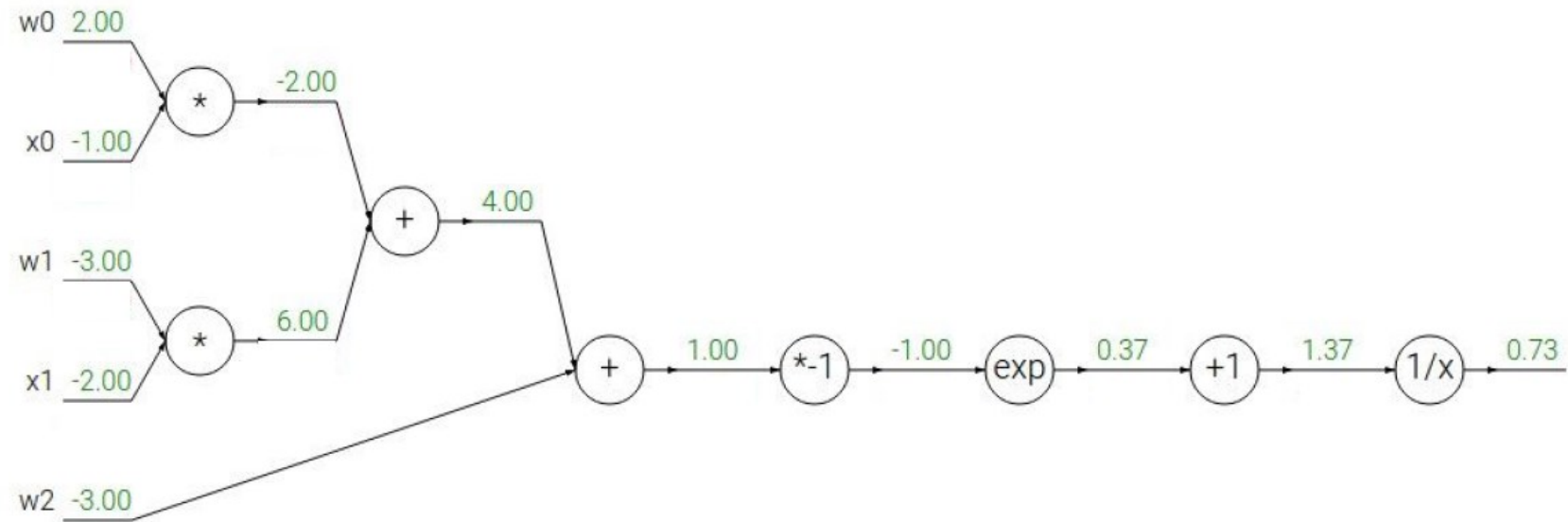
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Chain Rule(Local Gradient)



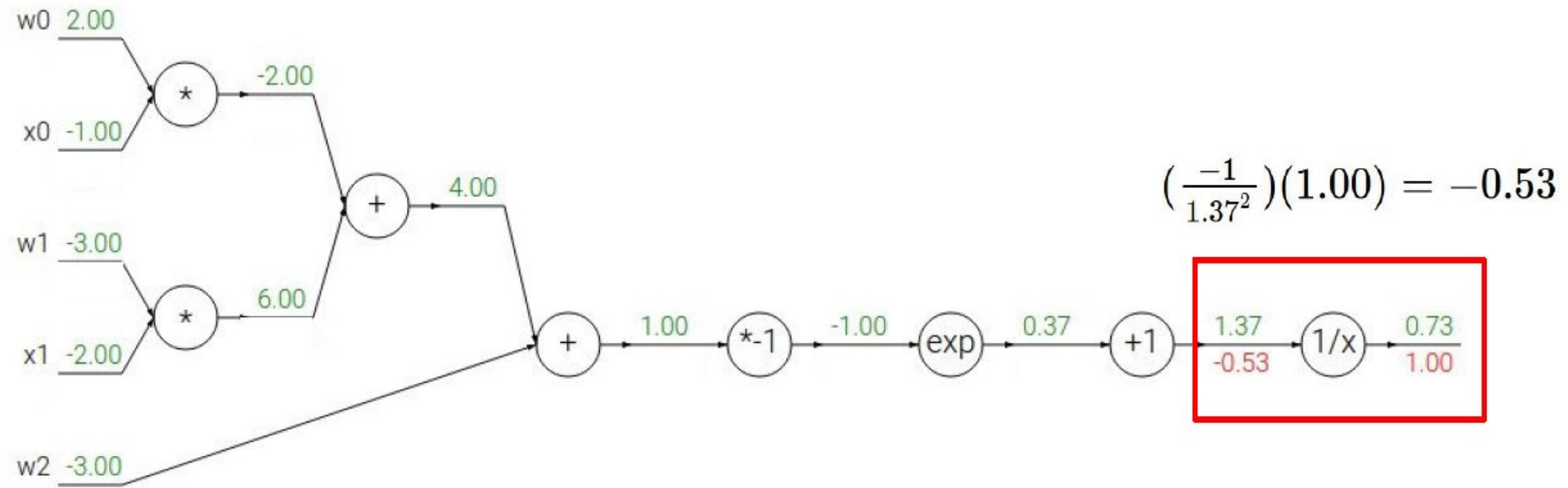
Back Propagation(Example)

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



Back Propagation(Example)

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



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

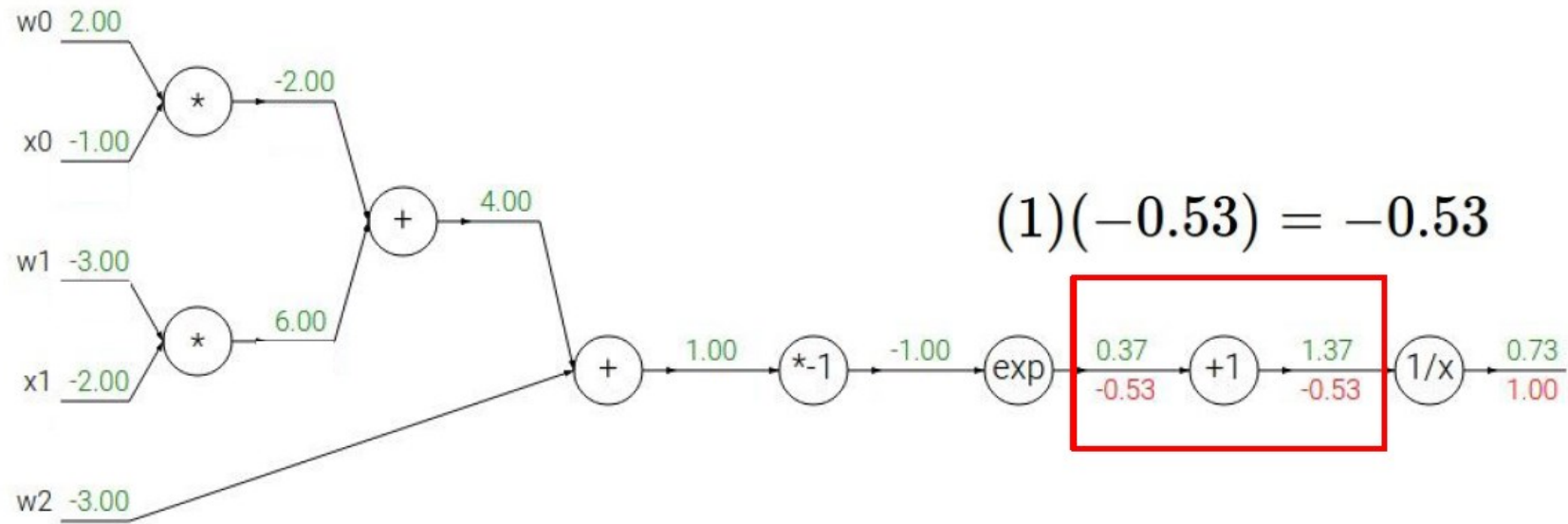
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Back Propagation(Example)

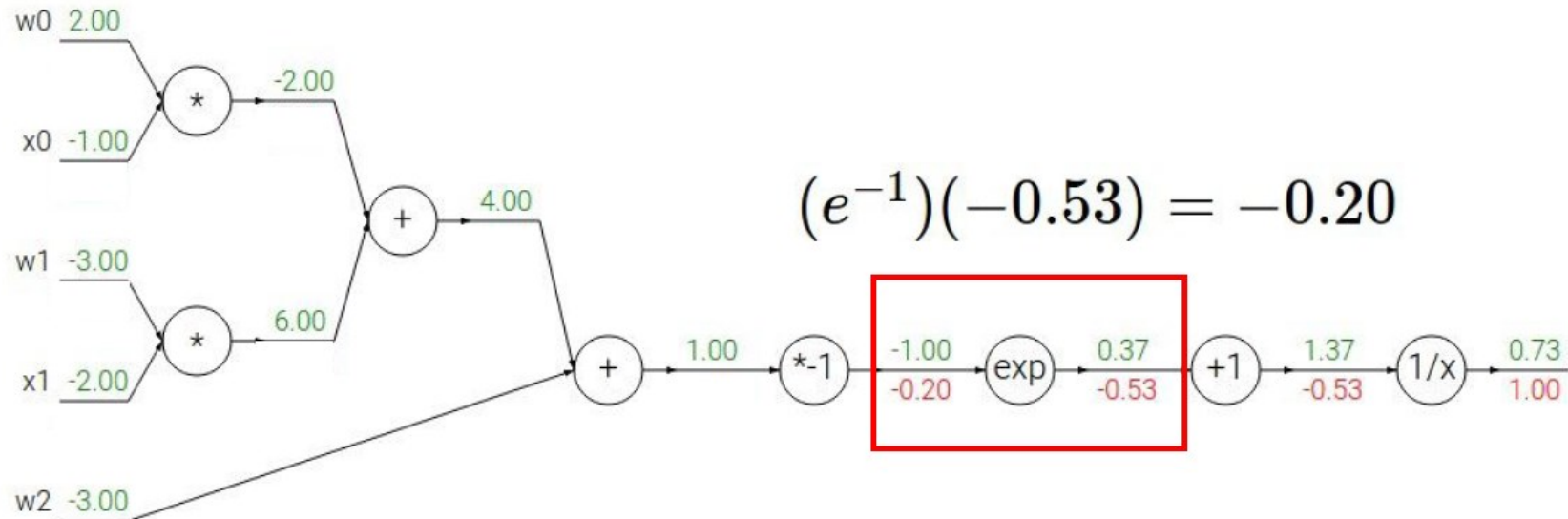
Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$f(x) = e^x$	\rightarrow	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	\rightarrow	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	\rightarrow	$\frac{df}{dx} = a$		$f_c(x) = c + x$	\rightarrow	$\frac{df}{dx} = 1$

Back Propagation(Example)

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

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

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

\rightarrow

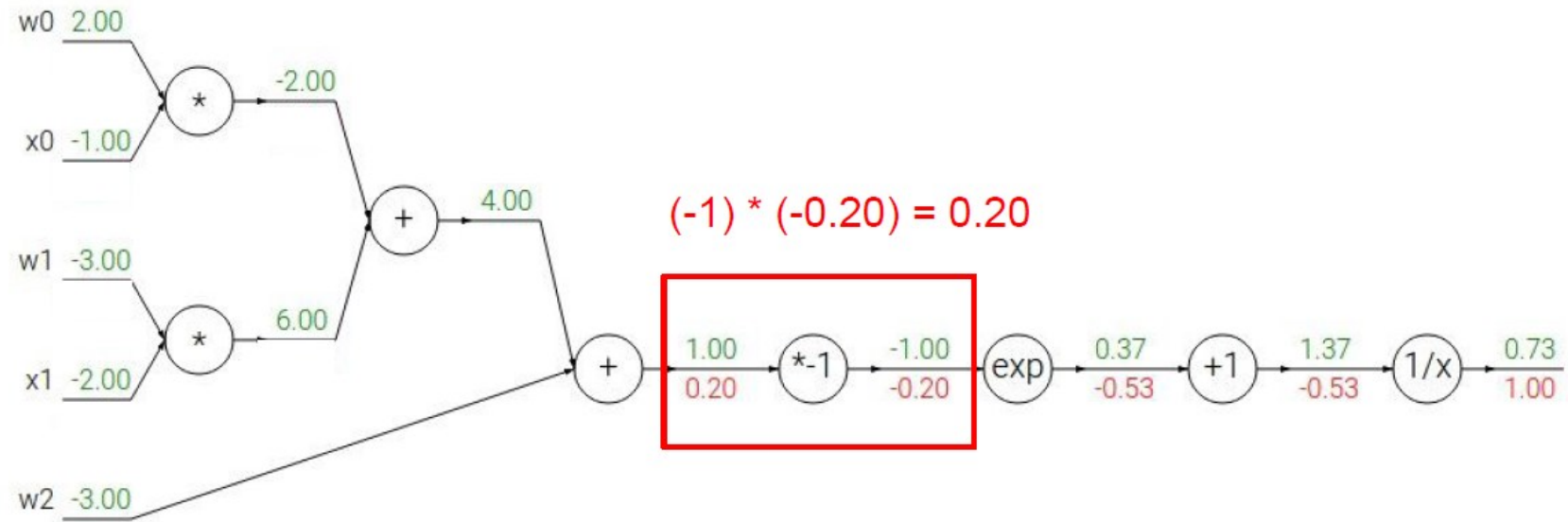
$$\frac{df}{dx} = -1/x^2$$

\rightarrow

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

Back Propagation(Example)

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

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

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

\rightarrow

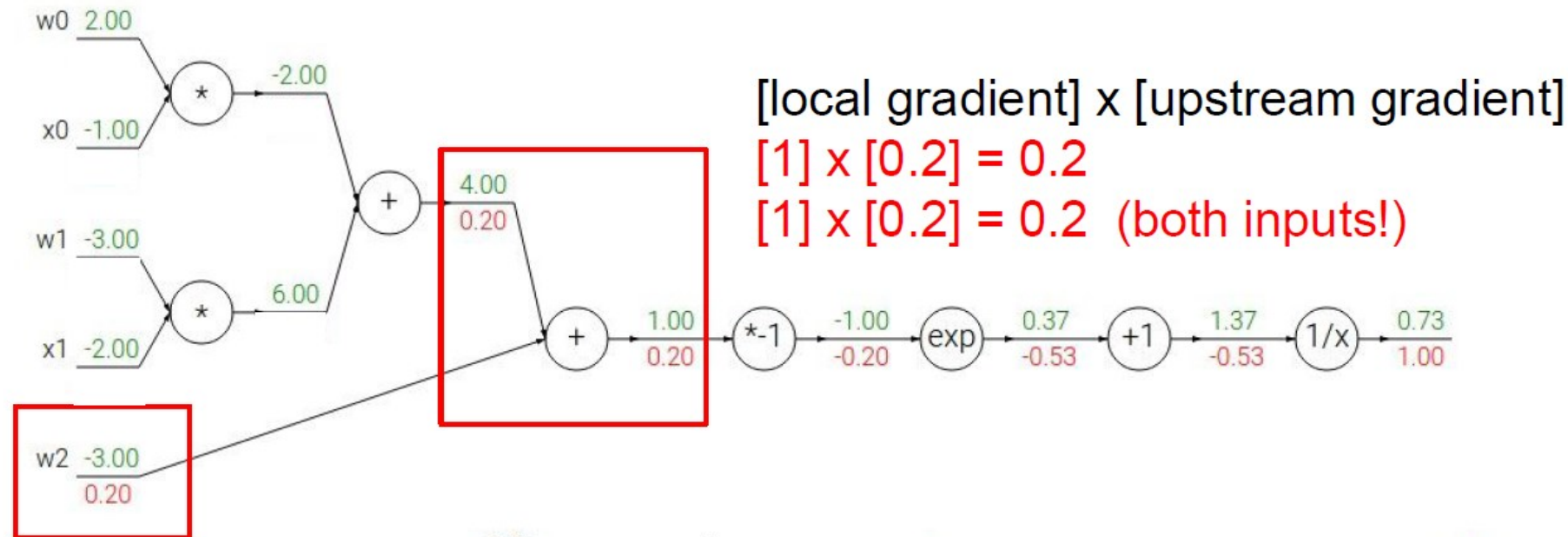
$$\frac{df}{dx} = -1/x^2$$

\rightarrow

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

Back Propagation(Example)

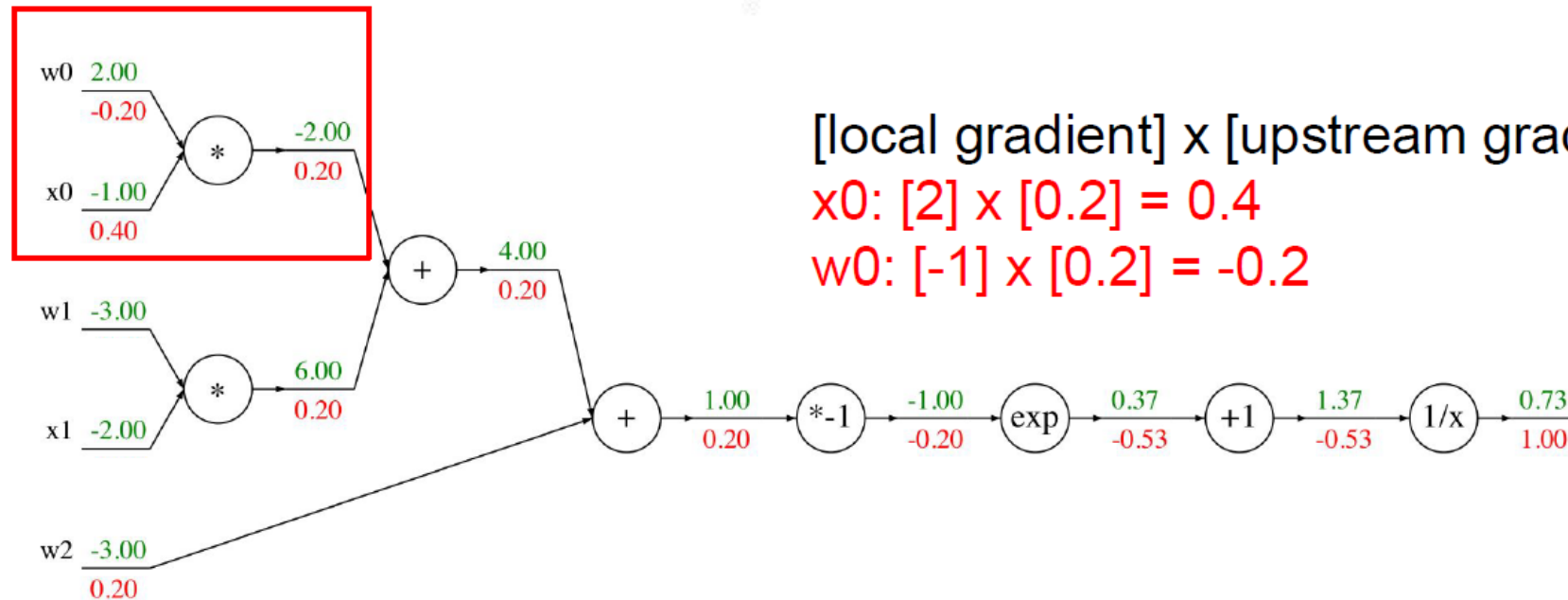
Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$



$f(x) = e^x$	\rightarrow	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	\rightarrow	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	\rightarrow	$\frac{df}{dx} = a$		$f_c(x) = c + x$	\rightarrow	$\frac{df}{dx} = 1$

Back Propagation(Example)

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$



[local gradient] x [upstream gradient]

$x_0: [2] \times [0.2] = 0.4$

$w_0: [-1] \times [0.2] = -0.2$

$f(x) = e^x$	\rightarrow	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	\rightarrow	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	\rightarrow	$\frac{df}{dx} = a$		$f_c(x) = c + x$	\rightarrow	$\frac{df}{dx} = 1$

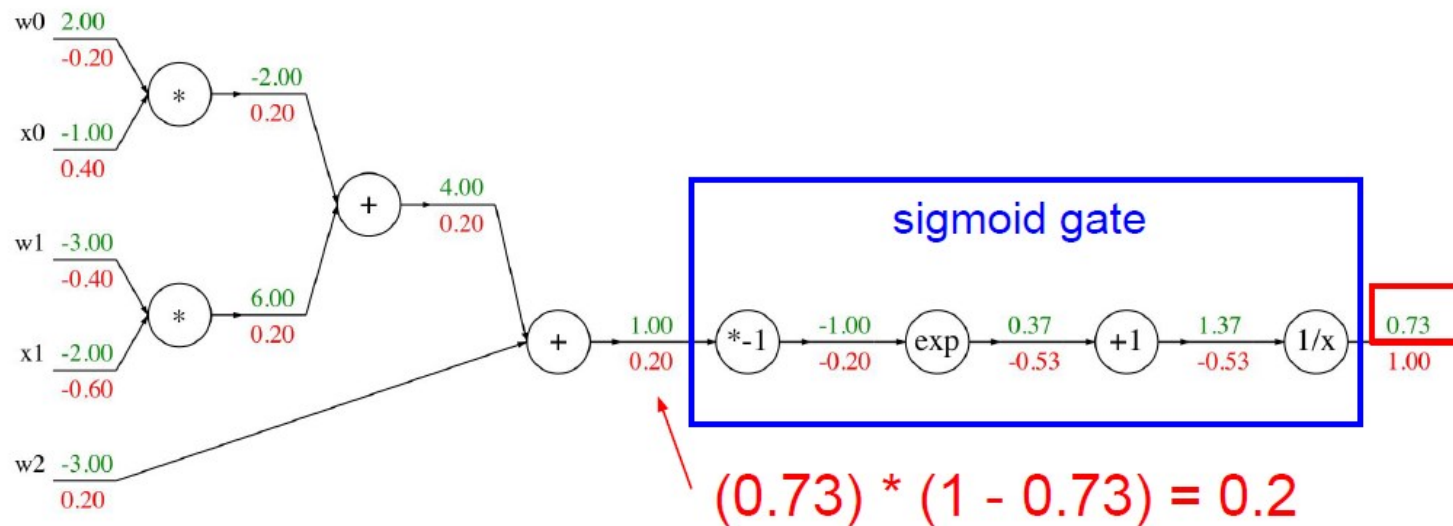
Back Propagation(Example)

$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2 x_2)}}$$

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

sigmoid function

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



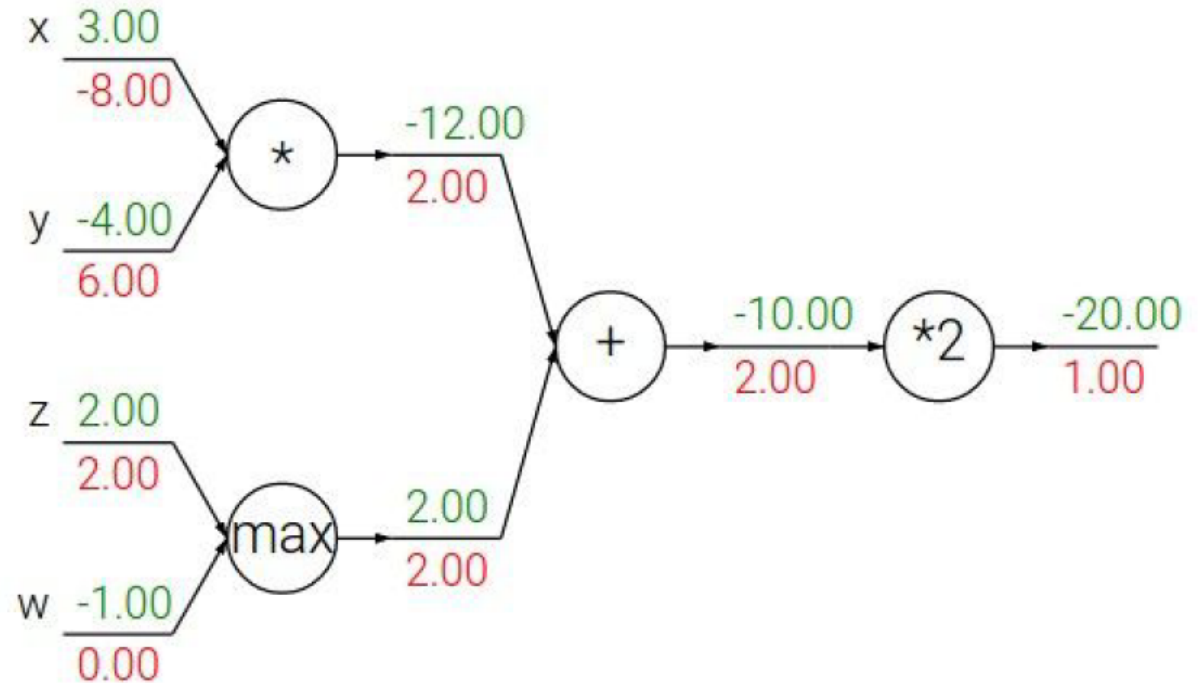
Back Propagation(Example)

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

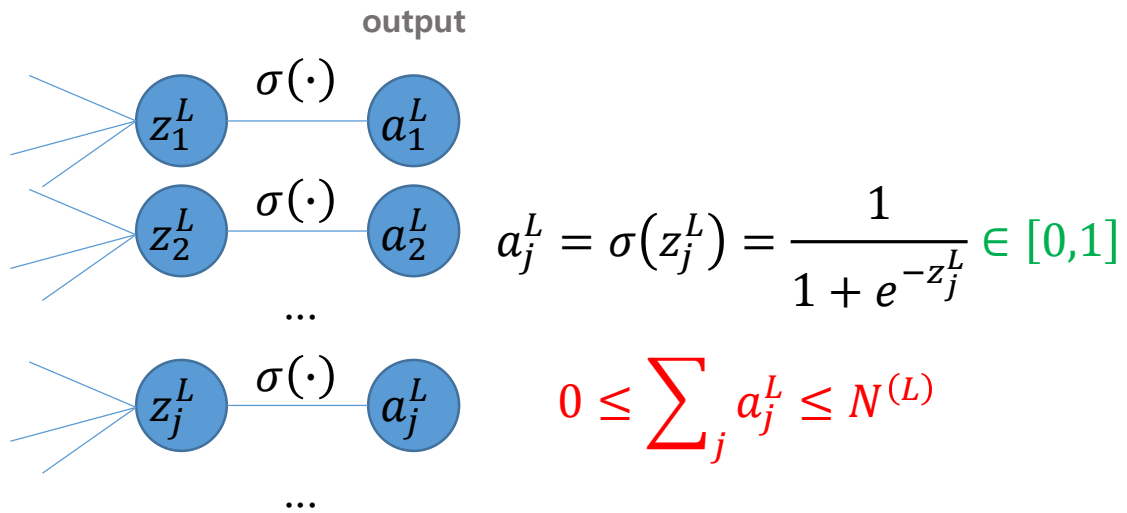
mul gate: gradient switcher



이제 Multi-Layer인 Network도 학습하는 방법은 알
았는데, 그럼 **Multi-Class**는?

Softmax!

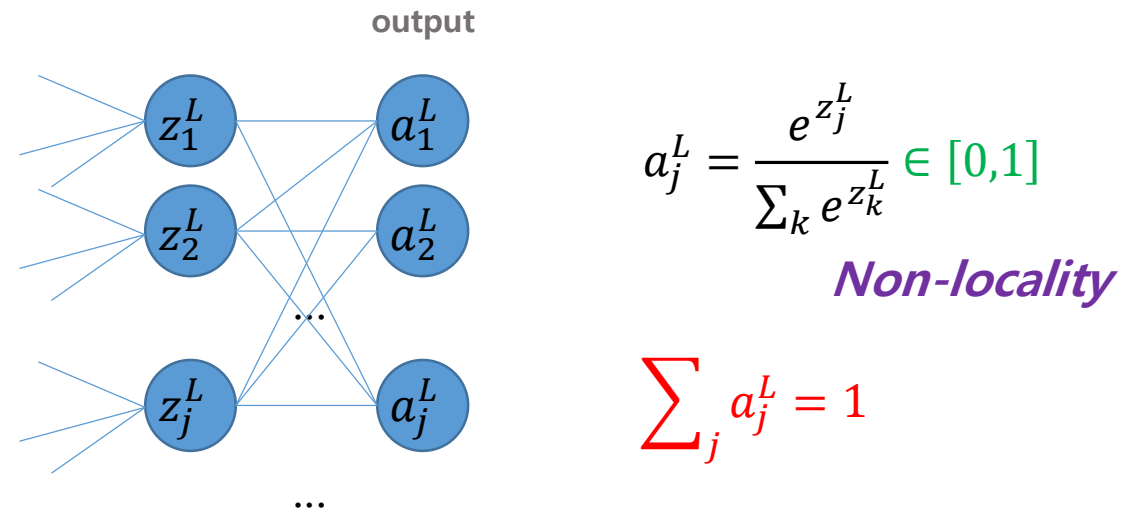
Original Output Layer



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

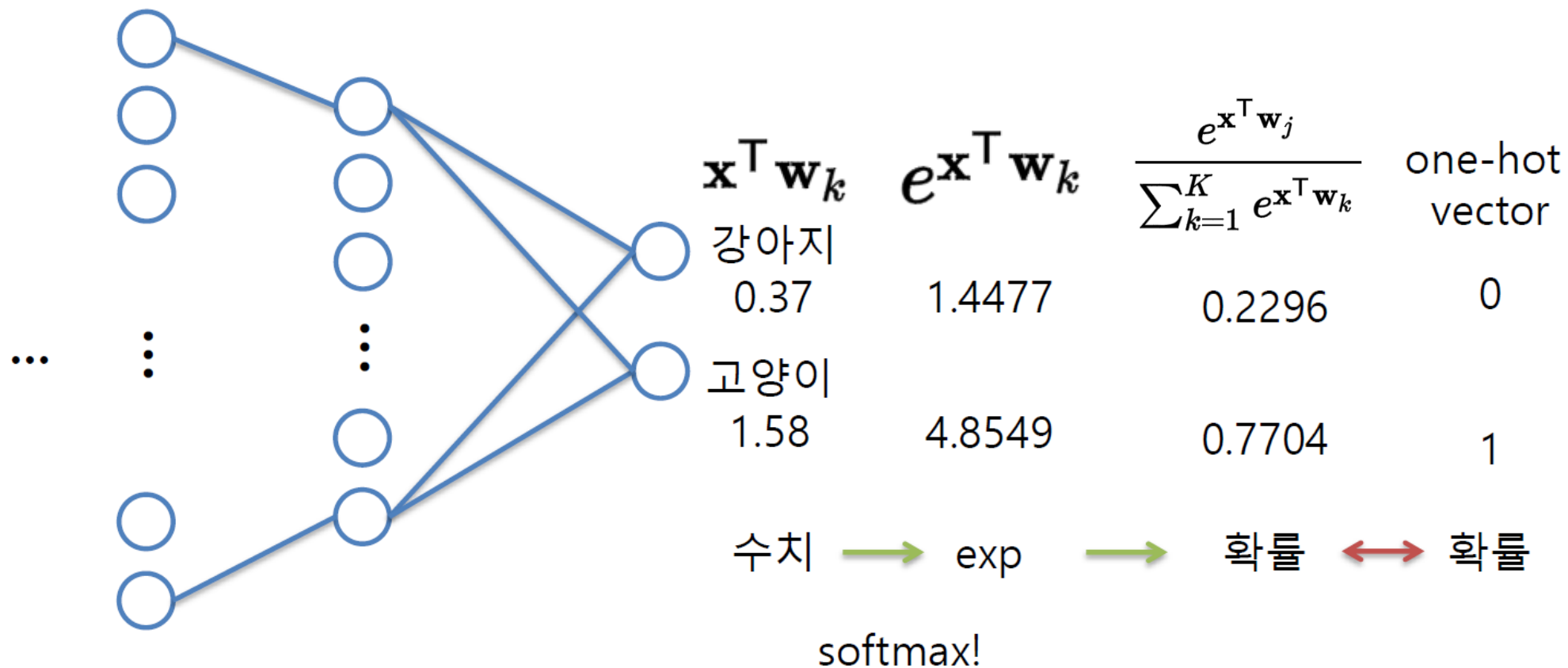


A softmax layer outputs a probability distribution!!

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

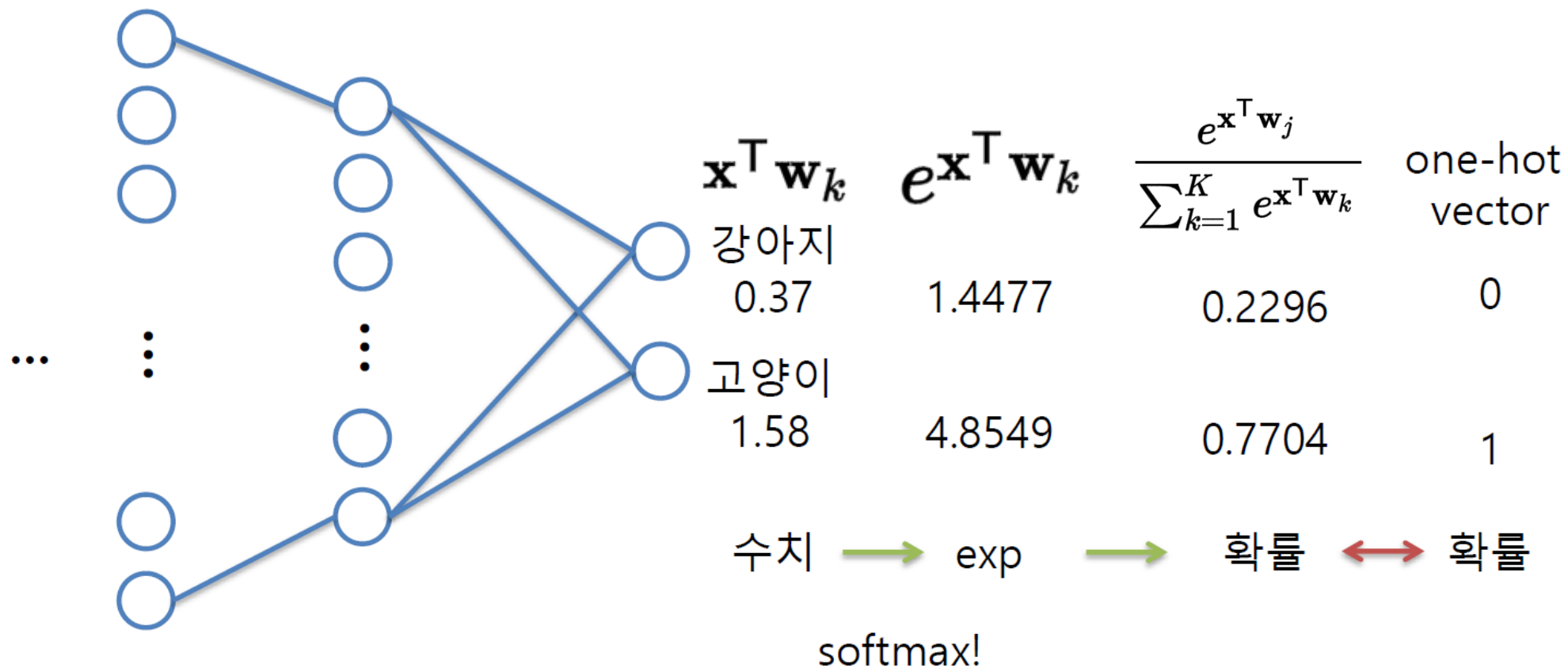
Softmax

- Output을 확률처럼 나타내보자 $P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^T \mathbf{w}_k}}$



Loss Function of Softmax

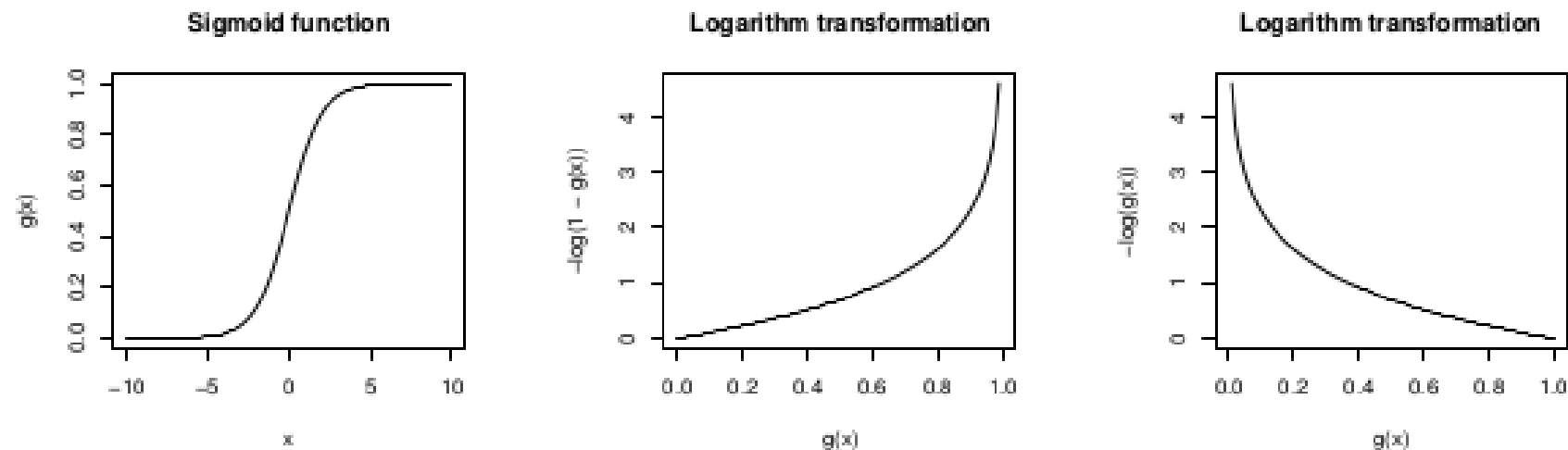
- Loss function은 어떻게 정의할까?
 - L1 loss or L2 loss(MSE)?



Loss Function of Logistic Regression

- Cross Entropy Loss!

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



(a) Sigmoid function.

(b) Cost for $y = 0$.

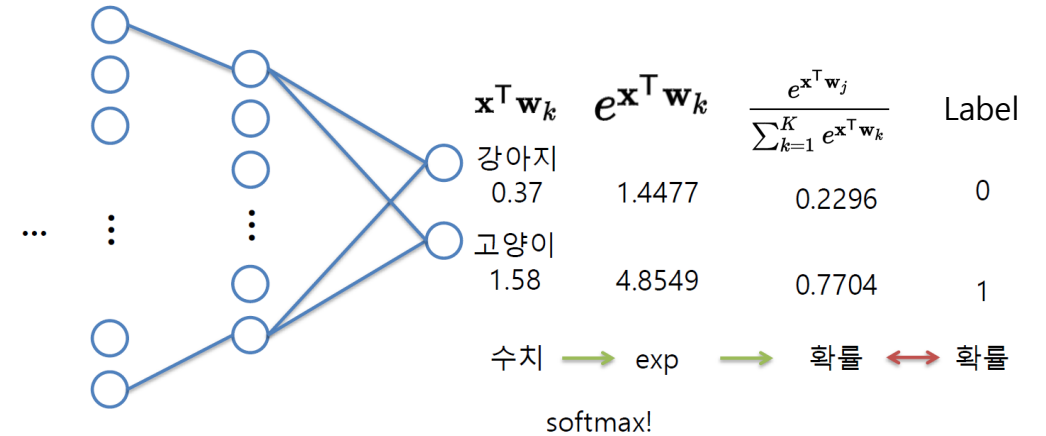
(c) Cost for $y = 1$.

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

Loss Function – Classification

- 마지막 layer의 activation function

- Softmax
$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^T \mathbf{w}_k}}$$



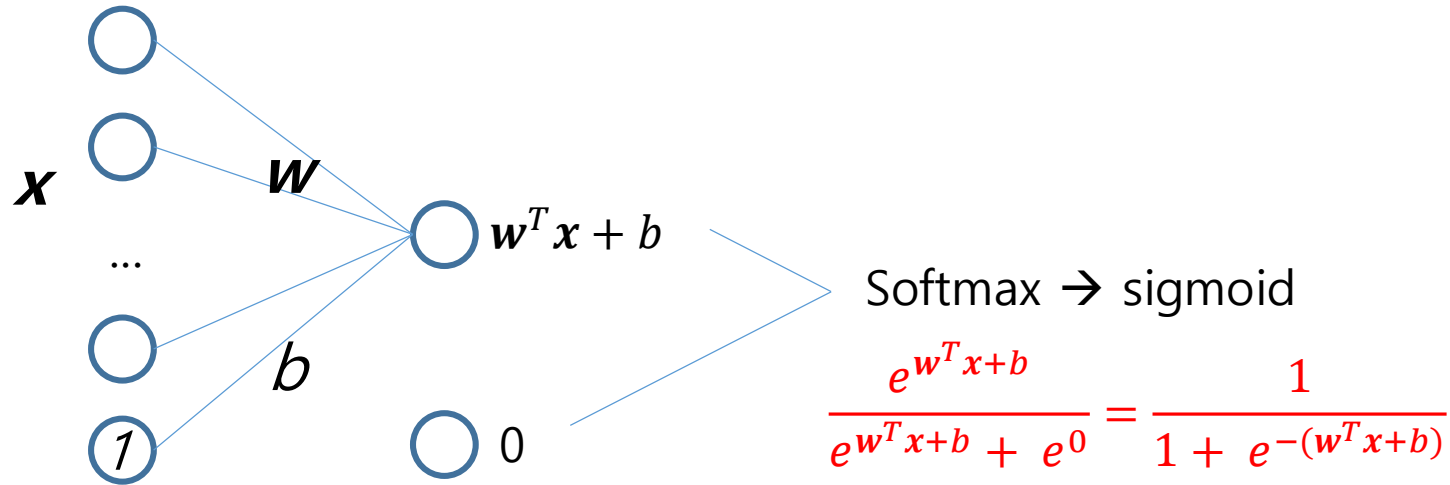
- Loss Function – Cross Entropy

- $$L = \sum -y \log y^*, \quad y \text{는 } label(\text{정답}), \quad y^* \text{는 } network \text{ output}(\text{예측값, softmax 결과})$$

Binary Classification

- 마지막 layer의 activation function

- Sigmoid – $H(x) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}} = P(y|\mathbf{x})$



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

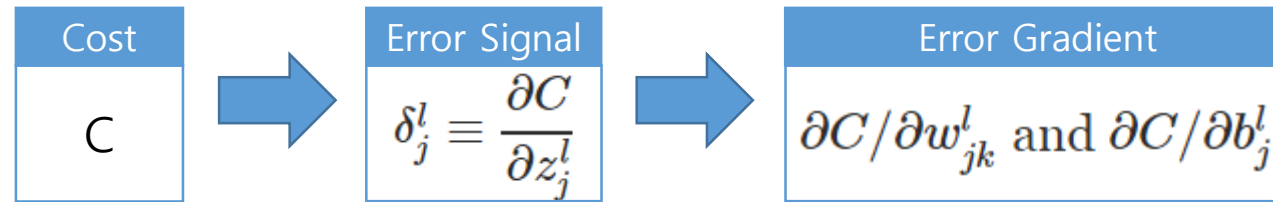
	Cross Entropy	Squared Loss
model1	$-1/3 \times (\log 0.4 + \log 0.4 + \log 0.1) = 1.38$	$1/3 \times (0.3^2 + 0.3^2 + (1-0.4)^2 + 0.3^2 + (1-0.4)^2 + 0.3^2 + (1-0.1)^2 + 0.2^2 + 0.7^2) = 0.81$
model2	$-1/3 \times (\log 0.7 + \log 0.7 + \log 0.4) = 0.64$	$1/3 \times (0.1^2 + 0.2^2 + (1-0.7)^2 + 0.1^2 +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

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

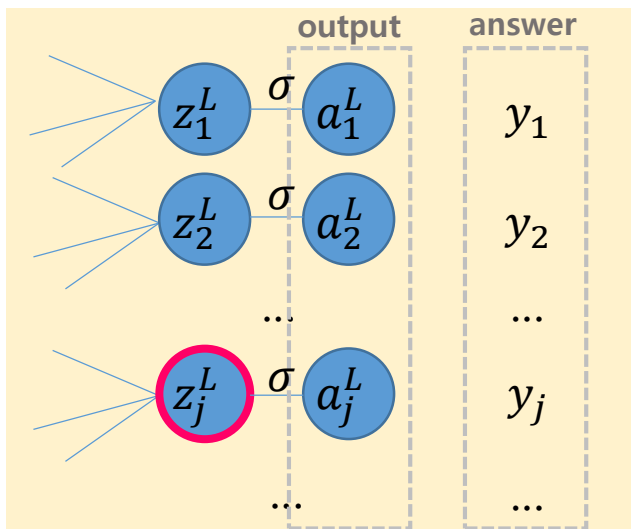


Equation

for the error in the output layer, δ^L

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

Chain rule



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

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

Only the j^{th} term is valid

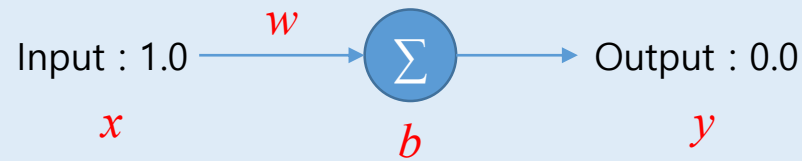
$$\frac{\partial C}{\partial z_j^L} = -(y_j - \sigma(z_j^L)) \sigma'(z_j^L) = (a_j^L - y_j) \sigma'(z_j^L)$$

Mean Squared Error

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

An example :

An Object



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

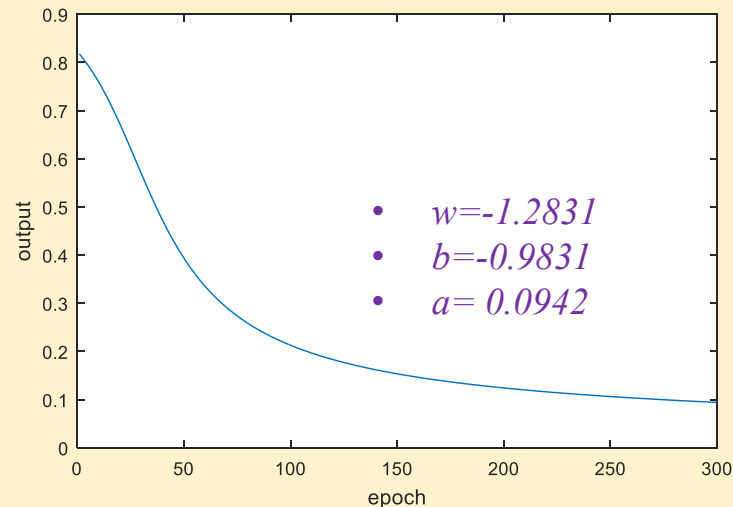
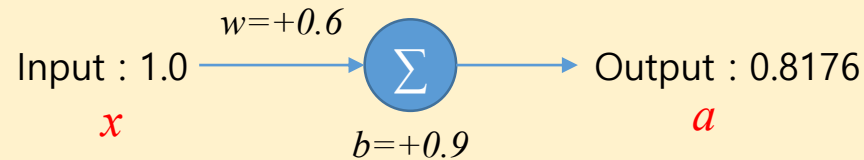
$$a = \sigma(z) = \sigma(wx + b) = \sigma(w + b)$$

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

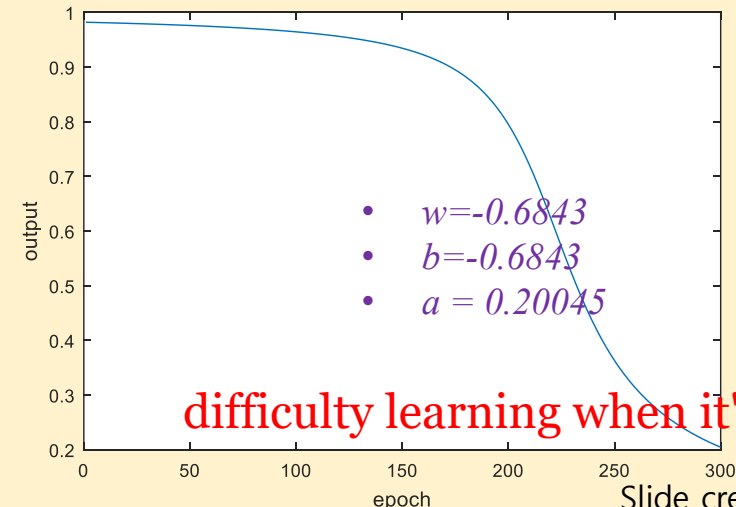
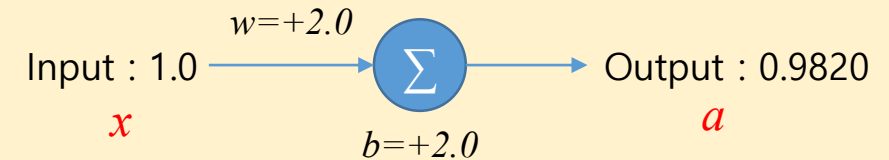
$$w = w - \eta\delta$$

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

Initialization 1



Initialization 2



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 :

An Object

Input : 1.0 \xrightarrow{w} Σ \xrightarrow{b} Output : 0.0

x b y

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

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

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

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

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

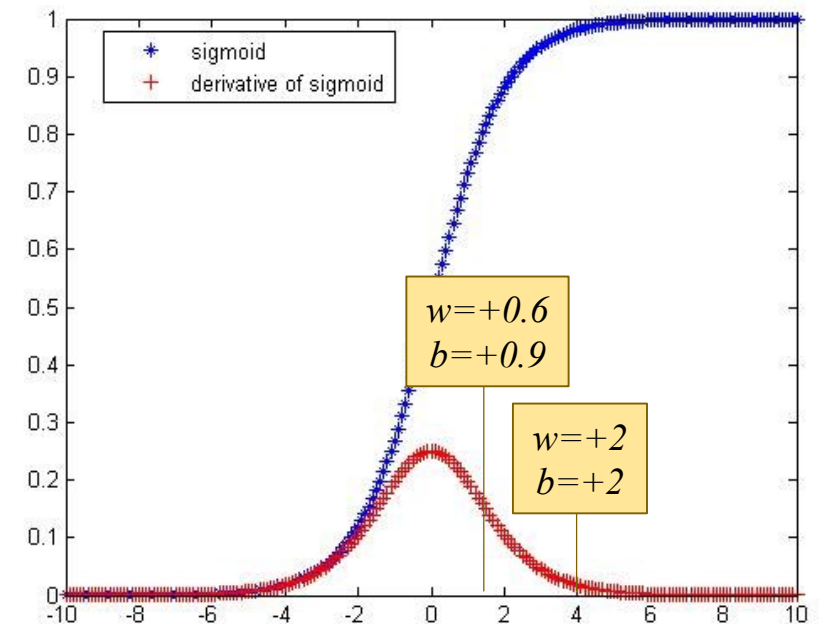
Learning slow means are $\partial C / \partial w$, $\partial C / \partial b$ small

Why they are small???

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

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

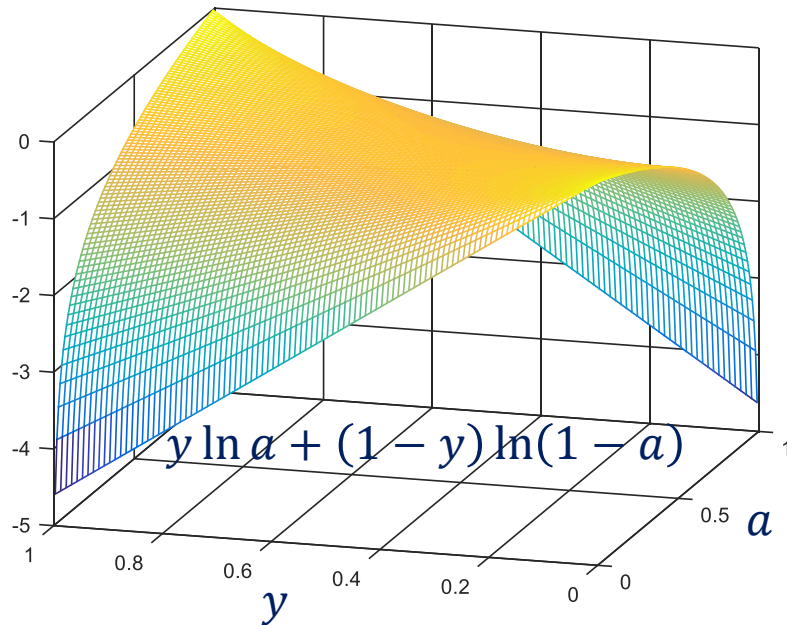
$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) = a\sigma'(z)$$



Cross Entropy Cost Function

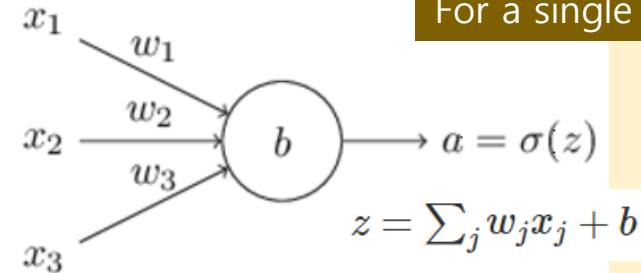
Introducing the cross-entropy cost function

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

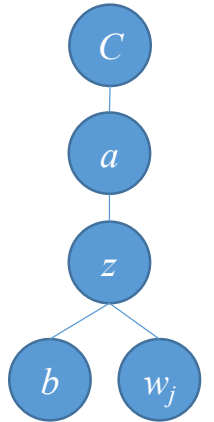


[Two features]

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



For a single neuron



Error gradient

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

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

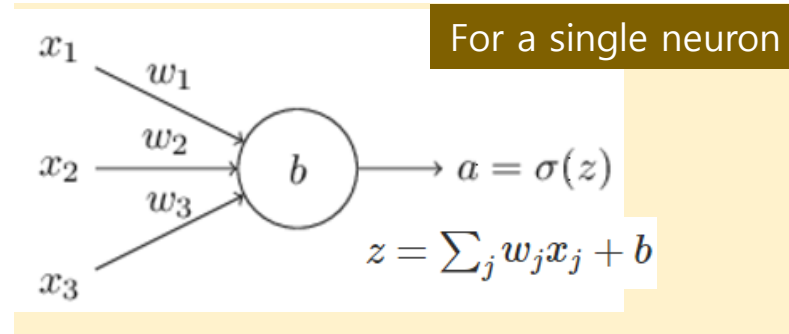
$$\frac{\partial a}{\partial z} = \sigma'(z) = (1 - \sigma(z))\sigma(z) = (1 - a)a \quad \frac{\partial z}{\partial w_j} = x_j$$

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

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

Cross Entropy vs MSE

Error gradient comparison



Cross – entropy cost

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

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

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

Quadratic cost

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

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

Error signal decreaseer!!!

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