Multi-**L**ayer **P**erceptron

Multi-Layer Perceptron

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

Network을 deep하게 쌓고, class도 여러 개일 때는 어떻게 학습할 수 있을까?

먼저 **Multi-Layer**부터 생각해봅시다

미분을 계산해봅시다!

$z11 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + x4 \cdot w14$

$a11 = \sigma(z11) = \dfrac{1}{1 + e^{-z11}}$

$z2 = a11 \cdot w21 + a12 \cdot w22 + a13 \cdot w23$

$a2 = z2$

$L = (y - a2)^2$

일 때,

w11을 update 하기 위해 필요한 미분값

$\dfrac{\partial L}{\partial w11} = ??????$

Back Propagation

$$z11 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + x4 \cdot w14$$

$$a11 = \sigma(z11) = \frac{1}{1 + e^{-z11}}$$
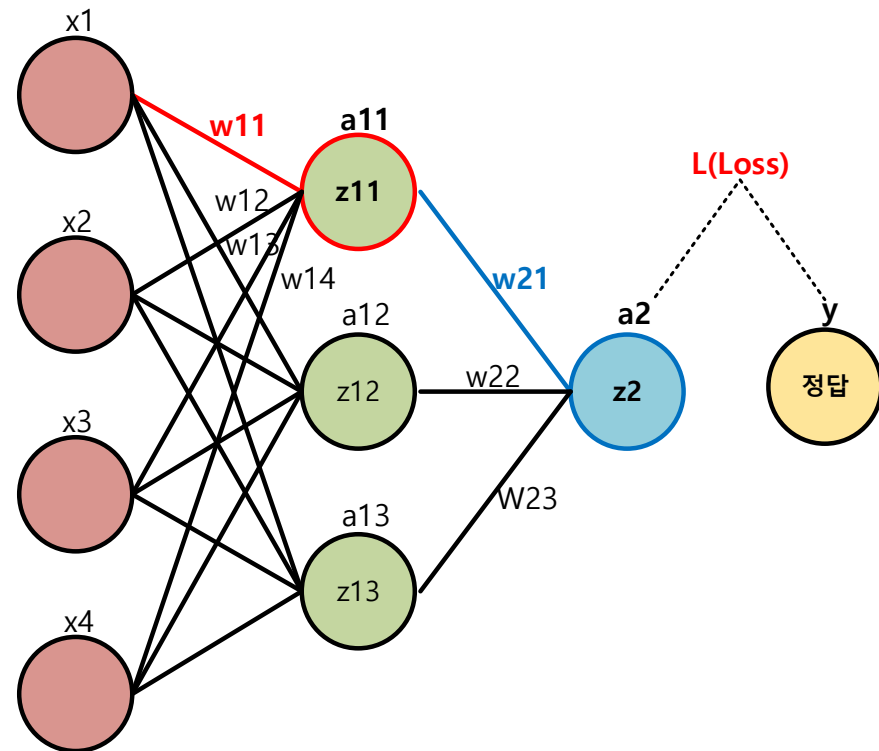
$$z2 = a11 \cdot w21 + a12 \cdot w22 + a13 \cdot w23$$

$$a2 = z2$$

$$L = (y - a2)^2$$

Loss부터 거꾸로 한 단계씩 미분을 해봅시다

$$\partial L / \partial a2 = -2(y - a2)$$

$$\partial a2 / \partial z2 = 1$$

$$\partial z2 / \partial a11 = w21$$

$$\partial a11 / \partial z11 = \sigma(z11) \cdot (1 - \sigma(z11))$$

$$\partial z11 / \partial w11 = x1$$

이 미분들을 전부 각각 곱하면(chain rule),

$$\frac{\partial L}{\partial a2} \cdot \frac{\partial a2}{\partial z2} \cdot \frac{\partial z2}{\partial a11} \cdot \frac{\partial a11}{\partial z11} \cdot \frac{\partial z11}{\partial w11} = \boldsymbol{\frac{\partial L}{\partial w11}}$$

우리가 구하려고 했던 미분값

x1 x2 x3 x4

w11 w12 w13 w14

a11 z11

a12 z12

a13 z13

w21 w22 W23

a2 z2

w21

L(Loss)

y 정답
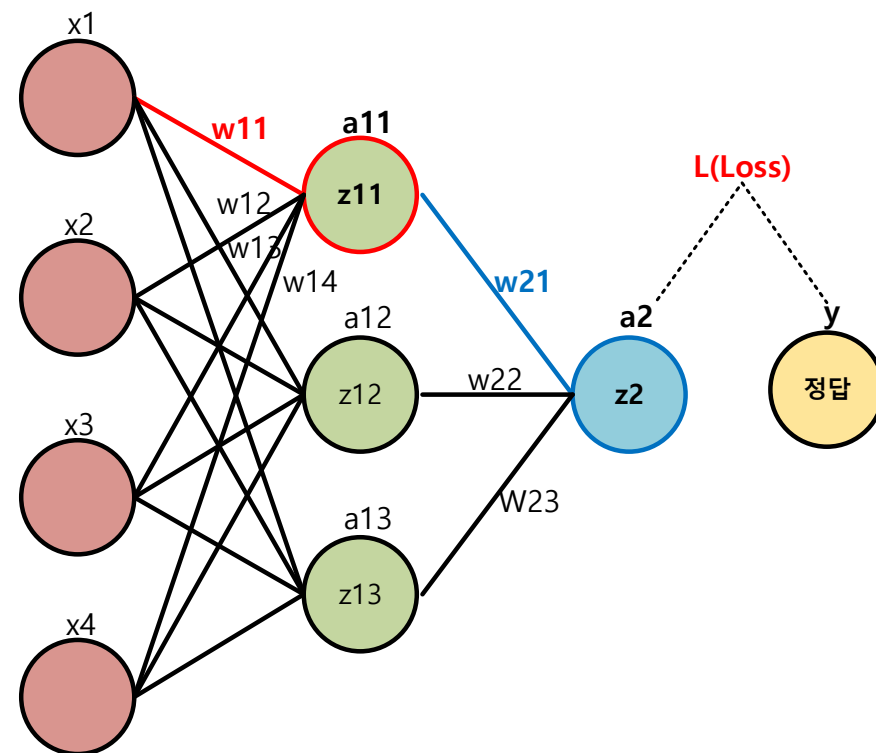
Back Propagation

Loss로부터 거꾸로 한 단계씩 미분 값을 구하고 이 값들을 chain rule 에 의하여 곱해가면서 weight에 대한 gradient를 구하는 방법
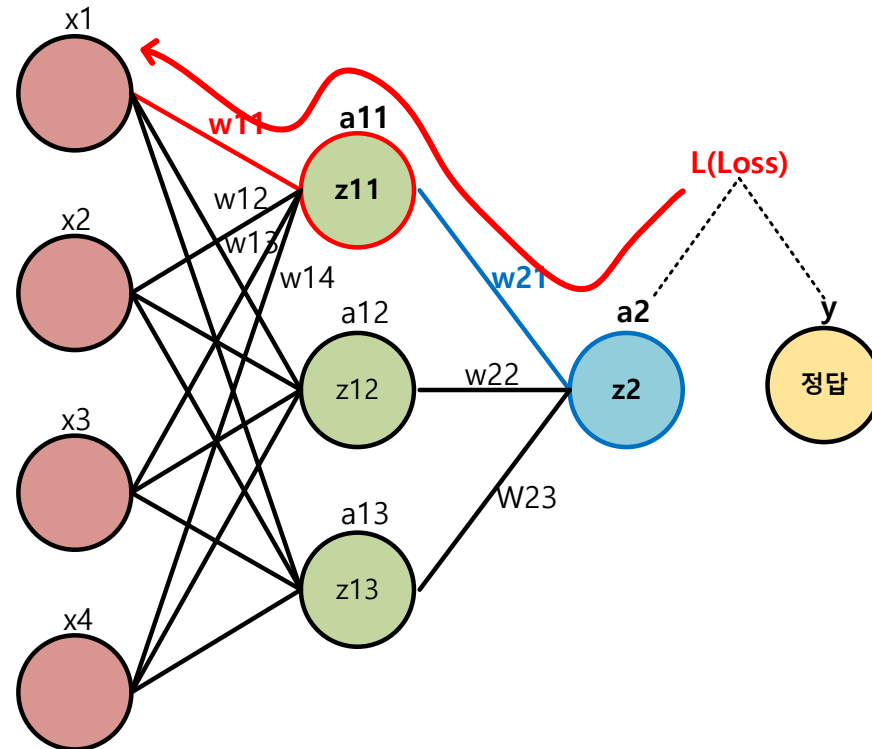
Back Propagation
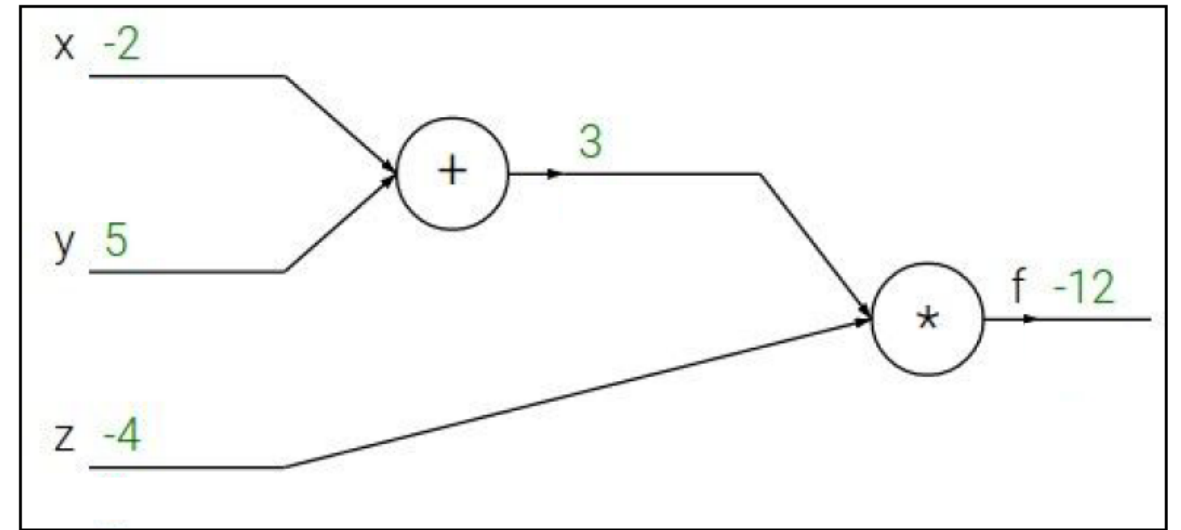
Backpropagation: a simple example

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

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

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

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

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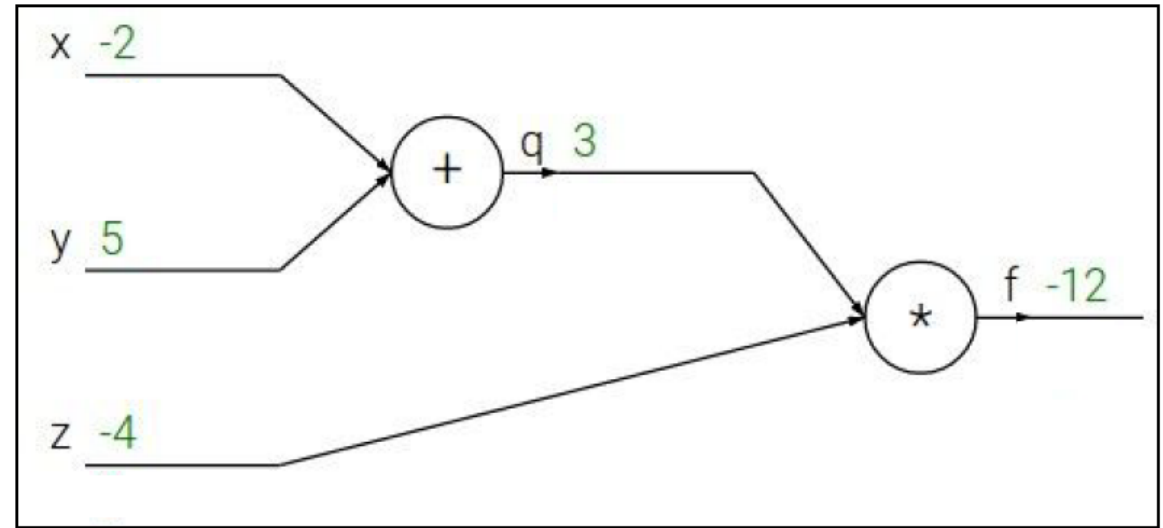
Back Propagation

Backpropagation: a simple example

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

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

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

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

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

x -2
   -4

y 5
   -4

q 3
   -4

z -4
   3

f -12
   1

+

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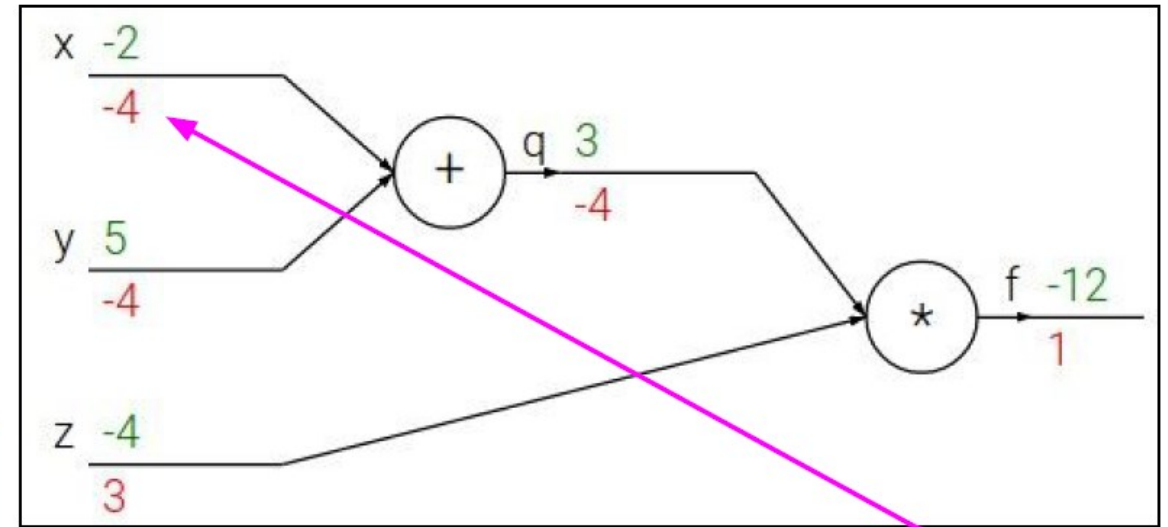
$\frac{\partial f}{\partial x}$

Chain rule:

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

Chain Rule(Local Gradient)
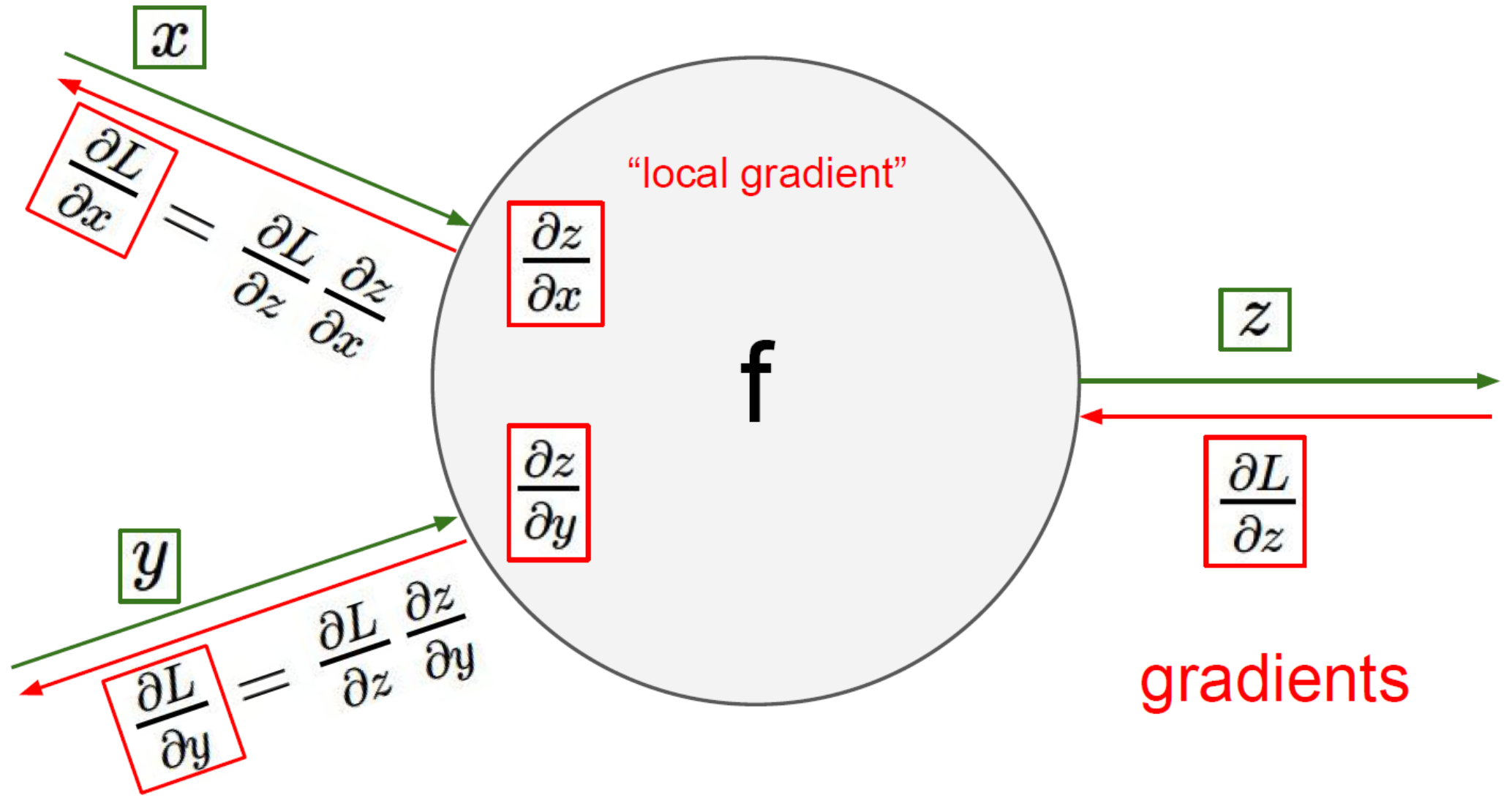
Back Propagation(Example)

Another example:

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

Back Propagation(Example)

Another example:

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

$$\left(\frac{-1}{1.37^2}\right)(1.00) = -0.53$$

w0 2.00

x0 -1.00

-2.00

w1 -3.00

x1 -2.00

6.00

4.00

1.00

w2 -3.00

1.37
-0.53

0.73
1.00

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

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

Back Propagation(Example)

Another example:

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

$$(1)(-0.53) = -0.53$$

$$f(x) = e^x \qquad \rightarrow \qquad \frac{df}{dx} = e^x \quad \bigg| \quad f(x) = \frac{1}{x} \qquad \rightarrow \qquad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \qquad \rightarrow \qquad \frac{df}{dx} = a \quad \bigg| \quad f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$

Back Propagation(Example)

Another example: $f(w, x) = \dfrac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$

$$(e^{-1})(-0.53) = -0.20$$

$$f(x) = e^x \qquad \rightarrow \qquad \dfrac{df}{dx} = e^x \qquad \Big| \qquad f(x) = \dfrac{1}{x} \qquad \rightarrow \qquad \dfrac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \qquad \rightarrow \qquad \dfrac{df}{dx} = a \qquad \Big| \qquad f_c(x) = c + x \qquad \rightarrow \qquad \dfrac{df}{dx} = 1$$

Back Propagation(Example)

Another example:

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

(-1) * (-0.20) = 0.20

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

$$f_a(x) = ax \qquad \rightarrow \qquad \frac{df}{dx} = a \qquad \bigg| \qquad f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$

Back Propagation(Example)

Another example:

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

[local gradient] x [upstream gradient]
[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)

$$f(x) = e^x \qquad \rightarrow \qquad \frac{df}{dx} = e^x \quad \Big| \quad f(x) = \frac{1}{x} \qquad \rightarrow \qquad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \qquad \rightarrow \qquad \frac{df}{dx} = a \quad \Big| \quad f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$
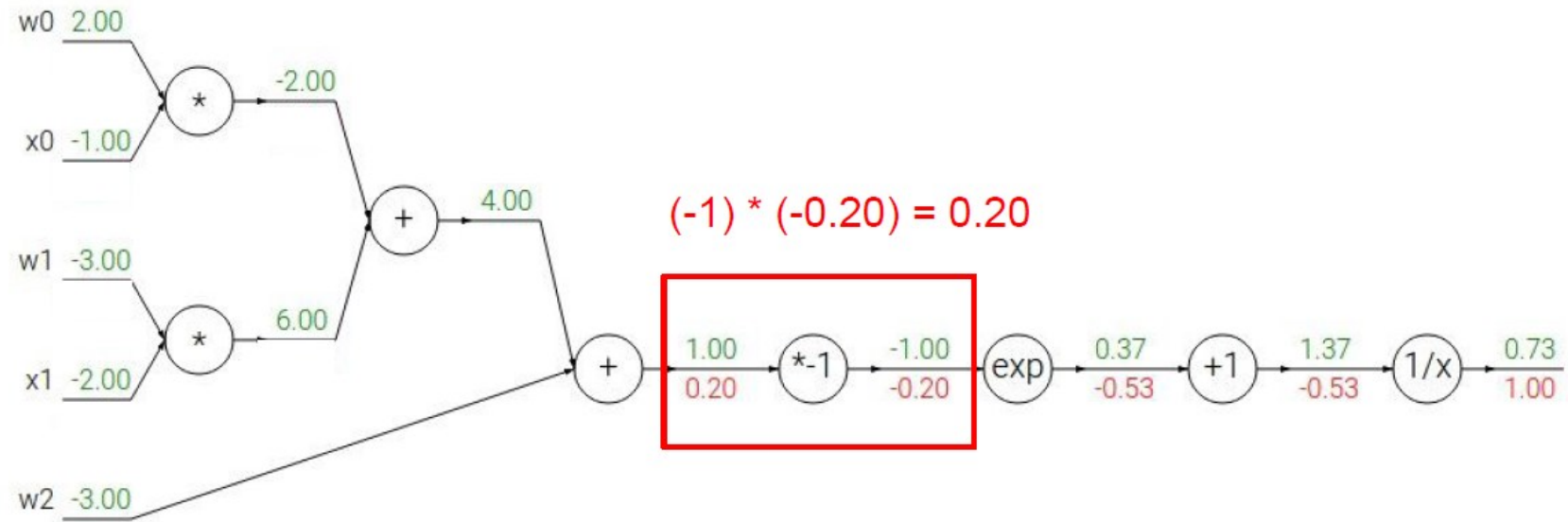
Back Propagation(Example)

Another example:

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

[local gradient] x [upstream gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

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

$$f_a(x) = ax \qquad \rightarrow \qquad \frac{df}{dx} = a \qquad \Bigg| \qquad f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$

Back Propagation(Example)

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

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

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

(0.73) * (1 - 0.73) = 0.2
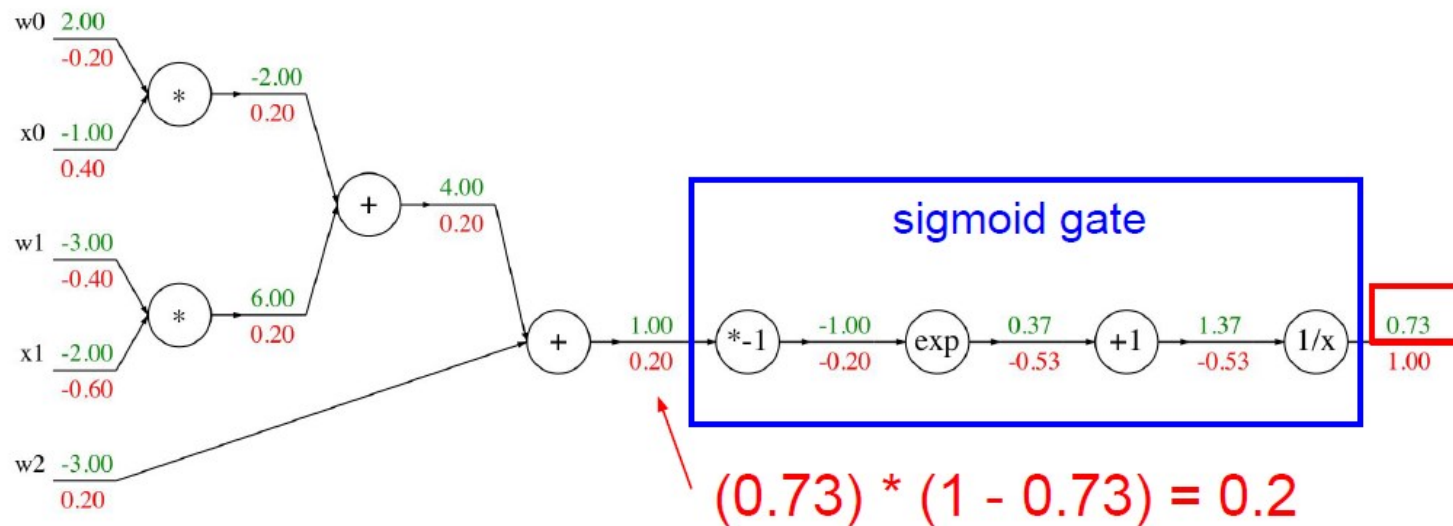
Back Propagation(Example)

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

mul gate: gradient switcher
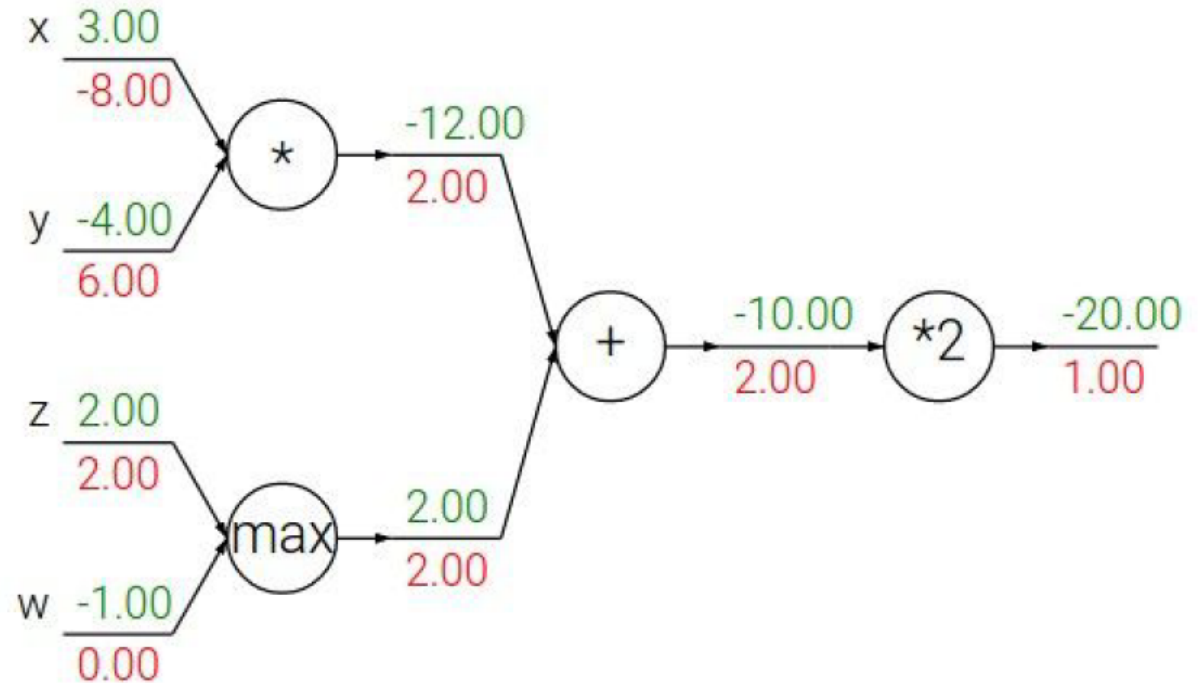
이제 Multi-Layer인 Network도 학습하는 방법은 알았는데, 그럼 **Multi-Class**는?

Softmax!

Original Output Layer	Output Layer of Softmax

Original Output Layer

output

$\sigma(\cdot)$

z_1^L → a_1^L

$\sigma(\cdot)$

z_2^L → a_2^L

...

$\sigma(\cdot)$

z_j^L → a_j^L

...

$$a_j^L = \sigma(z_j^L) = \frac{1}{1 + e^{-z_j^L}} \in [0,1]$$

$$0 \leq \sum_j a_j^L \leq N^{(L)}$$

Output Layer of Softmax

output

z_1^L → a_1^L

z_2^L → a_2^L

...

z_j^L → a_j^L

...

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \in [0,1]$$

Non-locality

$$\sum_j a_j^L = 1$$
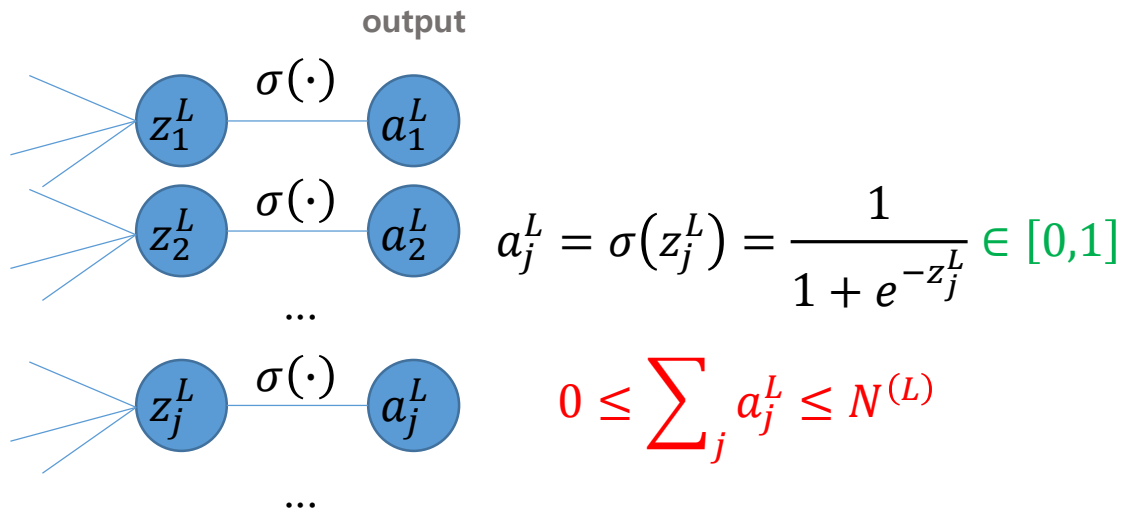
In classification problem, a desired output vector contains zero elements except only one '1' element .

→ This can be view as sum of all elements is 1
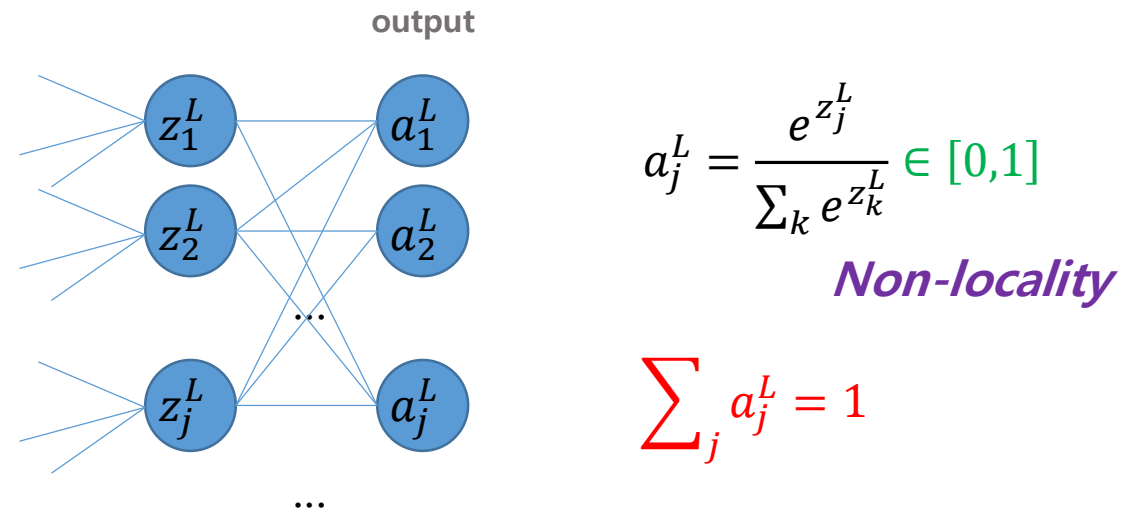
A softmax layer outputs a probability distribution!!

Monotonicity $\quad \dfrac{\partial a_j^L}{\partial z_k^L} = \underline{positive}$ if j=k, $\underline{negative}$ otherwise

Softmax

- Output을 확률처럼 나타내보자

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

	$\mathbf{x}^{\mathsf{T}}\mathbf{w}_k$	$e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}$	$\dfrac{e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}}$	one-hot vector
강아지	0.37	1.4477	0.2296	0
고양이	1.58	4.8549	0.7704	1

수치 ⟶ exp ⟶ 확률 ⟷ 확률
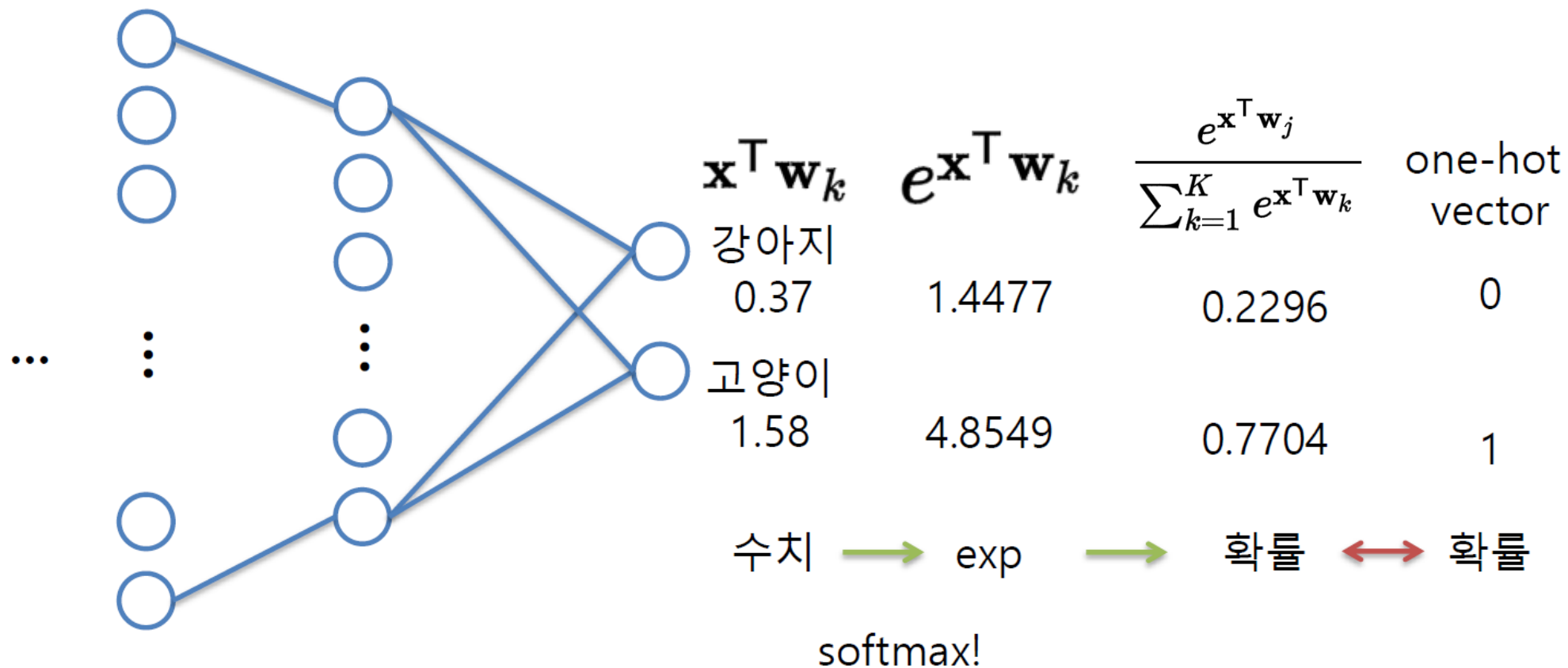
softmax!

Loss Function of Softmax

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

$$\mathbf{x}^{\mathsf{T}}\mathbf{w}_k \qquad e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k} \qquad \frac{e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}} \qquad \text{one-hot vector}$$

	$\mathbf{x}^{\mathsf{T}}\mathbf{w}_k$	$e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}$	softmax	one-hot vector
강아지	0.37	1.4477	0.2296	0
고양이	1.58	4.8549	0.7704	1

수치 ⟶ exp ⟶ 확률 ⟷ 확률

softmax!
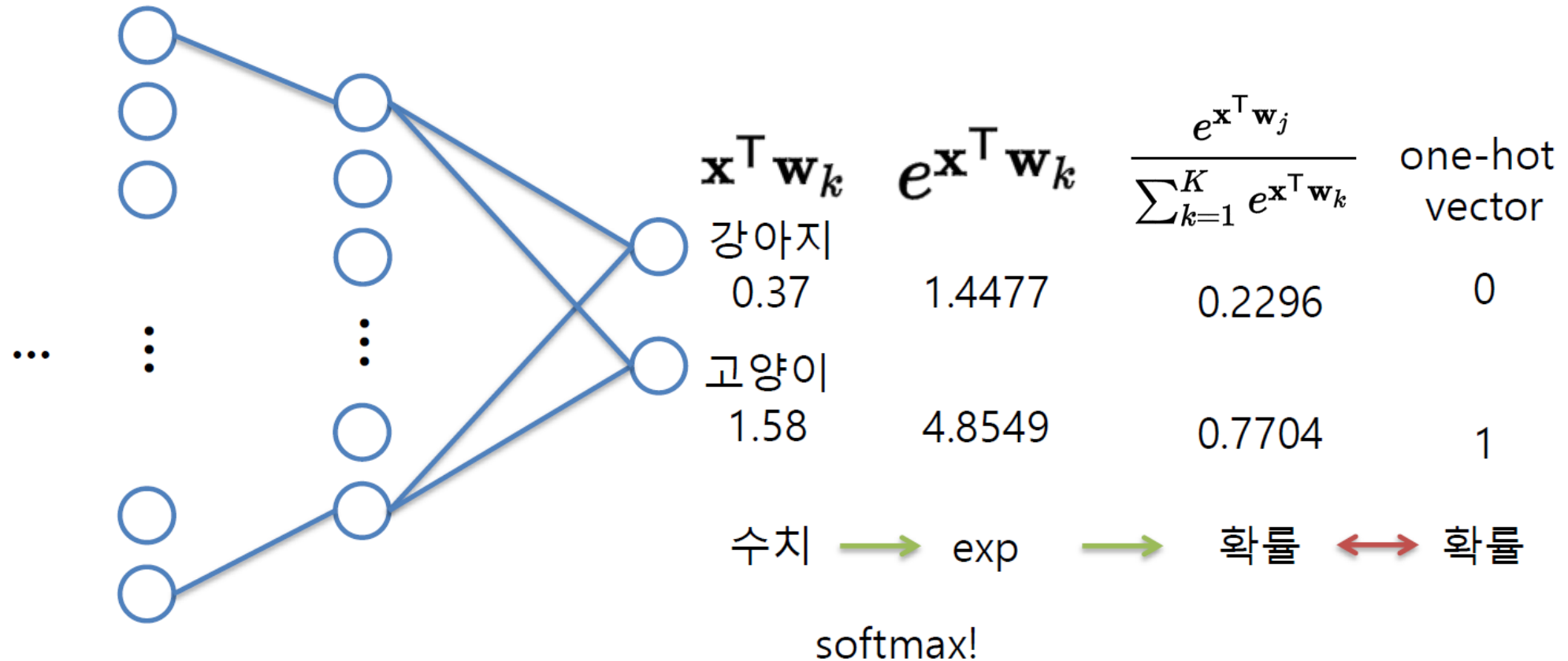
Loss Function of Logistic Regression

- Cross Entropy Loss!

$$cost(W) = -\frac{1}{m}\sum ylog(H(x)) + (1-y)log(1-H(x))$$
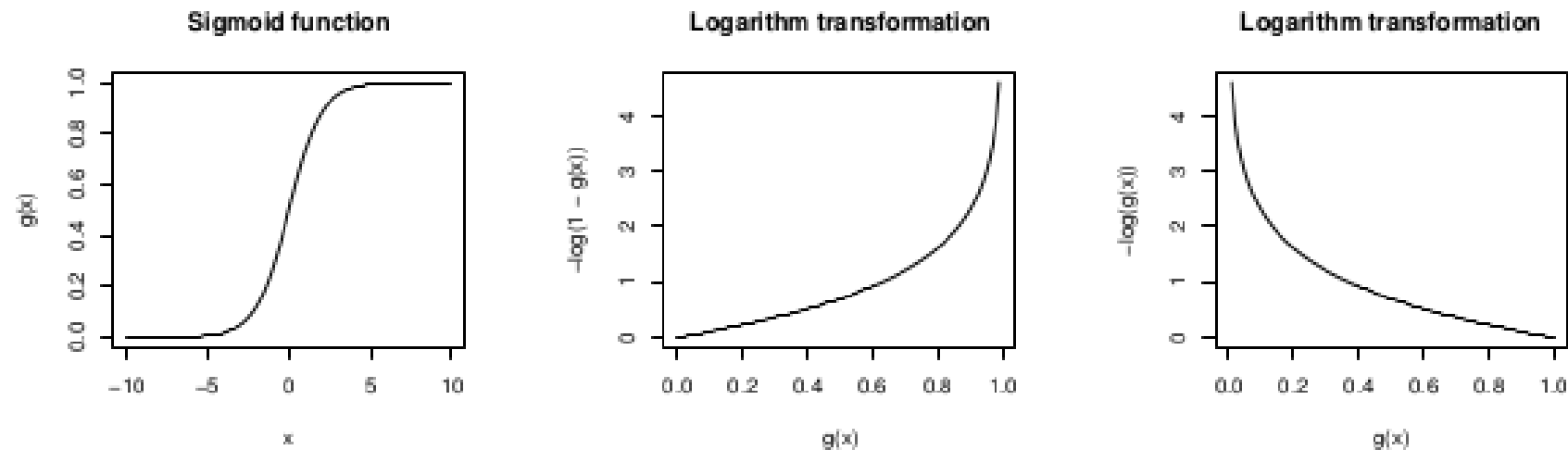
(a) *Sigmoid function.* (b) *Cost for $y=0$.* (c) *Cost for $y=1$.*

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

Loss Function – Classification

- 마지막 layer의 activation function

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

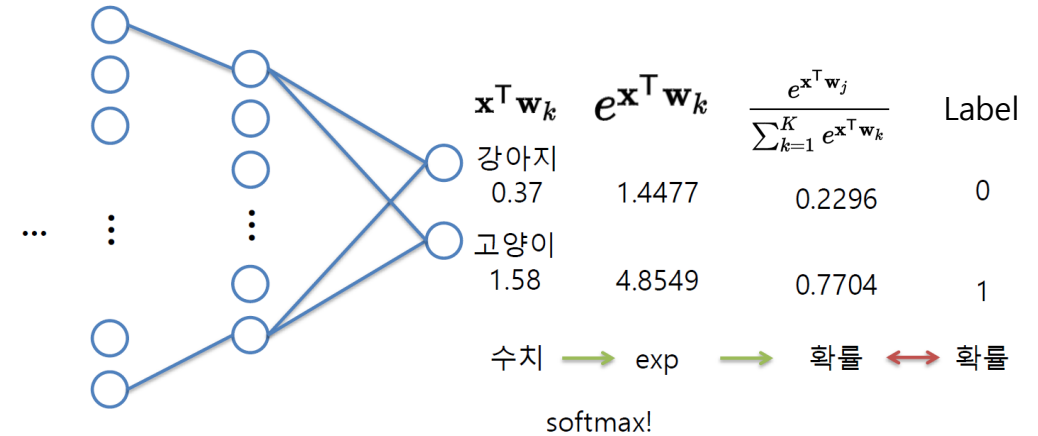|	$\mathbf{x}^\mathsf{T}\mathbf{w}_k$	$e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}$	$\frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$	Label
강아지	0.37	1.4477	0.2296	0
고양이	1.58	4.8549	0.7704	1

수치 ⟶ exp ⟶ 확률 ⟷ 확률

softmax!

- Loss Function – Cross Entropy
  - $L = \sum -y \log y^*,\ y$ 는 $label$(정답), y^* 는 $network\ output$(예측값, softmax 결과)

Binary Classification

- 마지막 layer의 activation function
  - Sigmoid $- H(x) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b) = \dfrac{1}{1 + e^{-(\boldsymbol{w}^T x + b)}} = P(y|\boldsymbol{x})$

$\boldsymbol{x}$

$\boldsymbol{w}$

...

b

$\boldsymbol{1}$

$\boldsymbol{w}^T \boldsymbol{x} + b$

0

Softmax → sigmoid

$$\frac{e^{\boldsymbol{w}^T x + b}}{e^{\boldsymbol{w}^T x + b} + e^0} = \frac{1}{1 + e^{-(\boldsymbol{w}^T x + b)}}$$

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 entrop와 squared loss에 따른 loss 값은?

Cross Entropy vs MSE

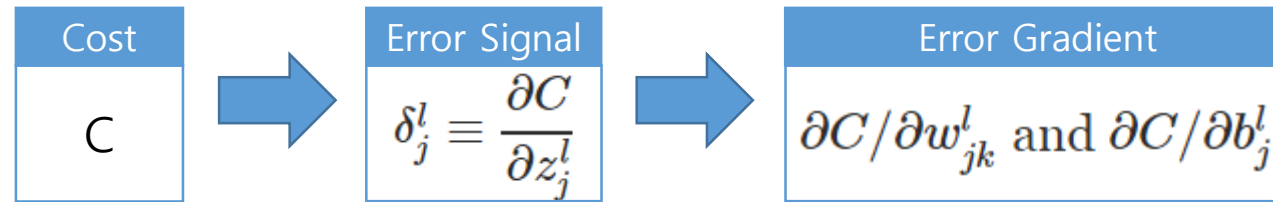|	Inference(softmax output)			True Label(onehot)			Correct?
	class1	class2	class3	class1	class2	class3	
model1	0.3	0.3	0.4	0	0	1	Yes
	0.3	0.4	0.3	0	1	0	Yes
	0.1	0.2	0.7	1	0	0	No
model2	0.1	0.2	0.7	0	0	1	Yes
	0.1	0.7	0.2	0	1	0	Yes
	0.3	0.4	0.3	1	0	0	No

	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
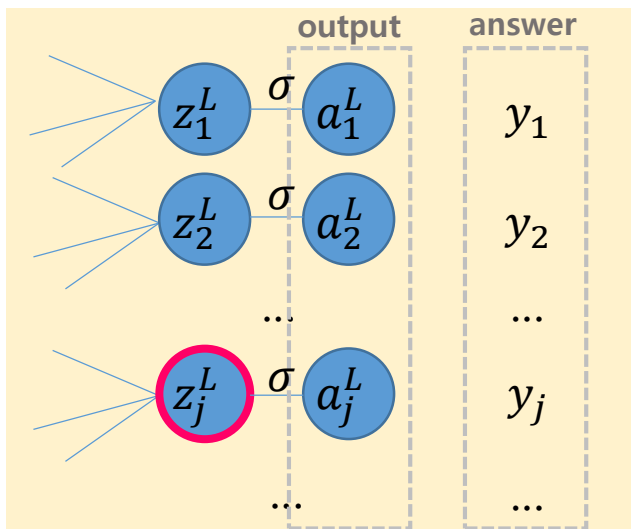
Cost
C

$\Rightarrow$

Error Signal
$\delta_j^l \equiv \dfrac{\partial C}{\partial z_j^l}$

$\Rightarrow$

Error Gradient
$\partial C / \partial w_{jk}^l$ and $\partial C / \partial b_j^l$

Equation **for the error in the output layer, δ^L**

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

Chain rule

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

output   answer

z_1^L σ a_1^L y_1

z_2^L σ a_2^L y_2

...   ...

z_j^L σ a_j^L y_j

...   ...

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

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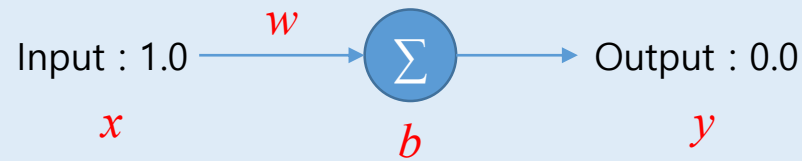
Only the jth term is valid

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

Slide credit : Hwalsuk Lee@CLAIR

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 $\longrightarrow$ Output : 0.0

x b y

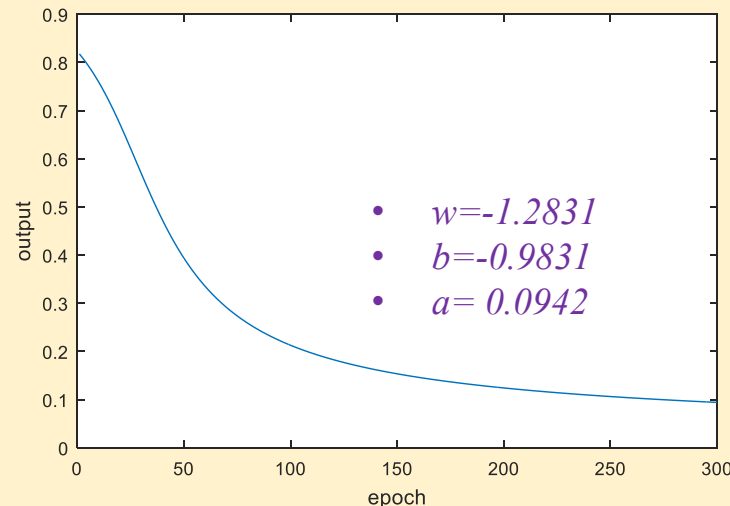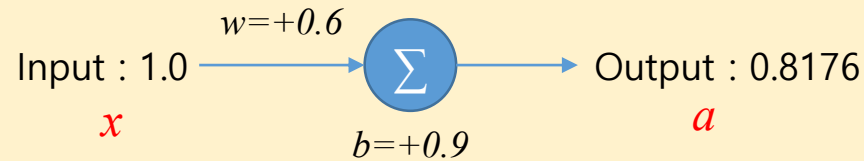$$C = \frac{(a - y)^2}{2} = \frac{a^2}{2}$$

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

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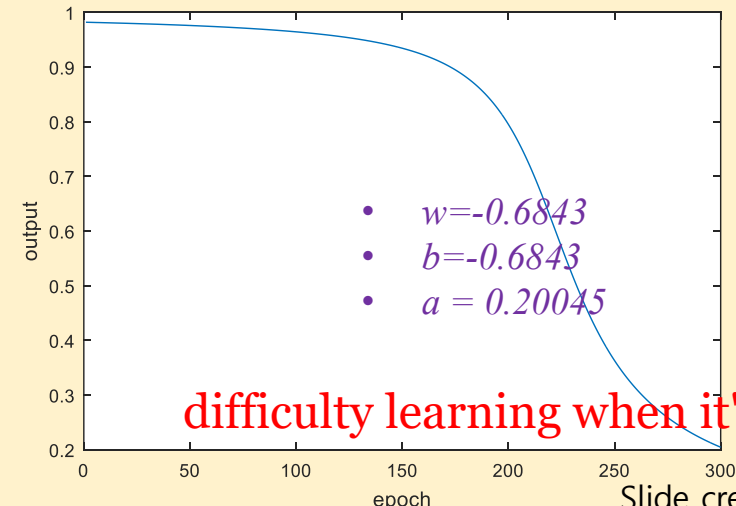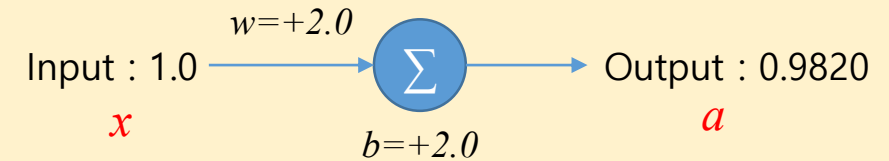$$b = b - \eta\delta$$

Initialization 1

$w=+0.6$

Input : 1.0 $\longrightarrow$ Σ $\longrightarrow$ Output : 0.8176

x $b=+0.9$ a

- $w=-1.2831$
- $b=-0.9831$
- $a= 0.0942$

Initialization 2

$w=+2.0$

Input : 1.0 $\longrightarrow$ Σ $\longrightarrow$ Output : 0.9820

x $b=+2.0$ a

- $w=-0.6843$
- $b=-0.6843$
- $a = 0.20045$

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 $\longrightarrow$ Output : 0.0

x $\quad\quad b$ $\quad\quad\quad y$

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

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

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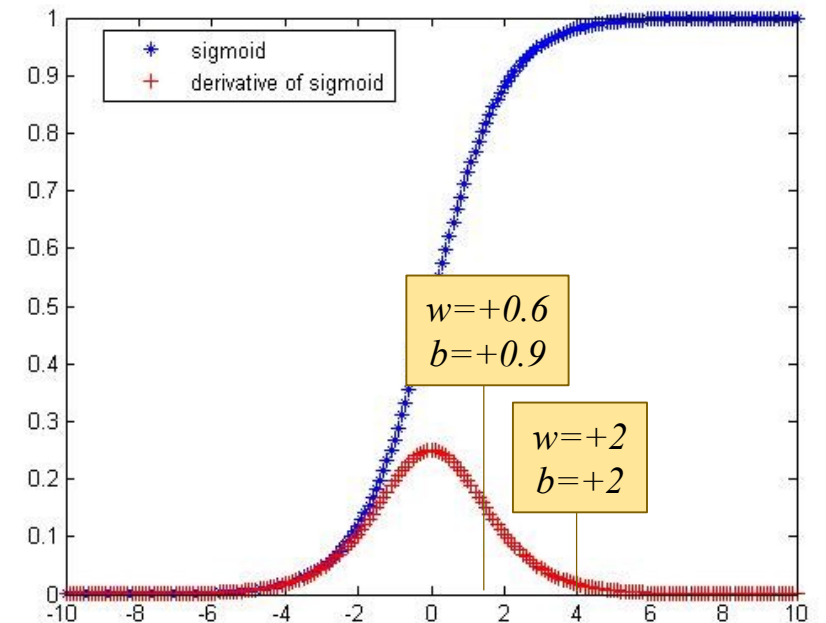
$$w = w - \eta\delta$$

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

Learning slow means are ∂C / ∂w, ∂C / ∂b small

Why they are small???

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

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

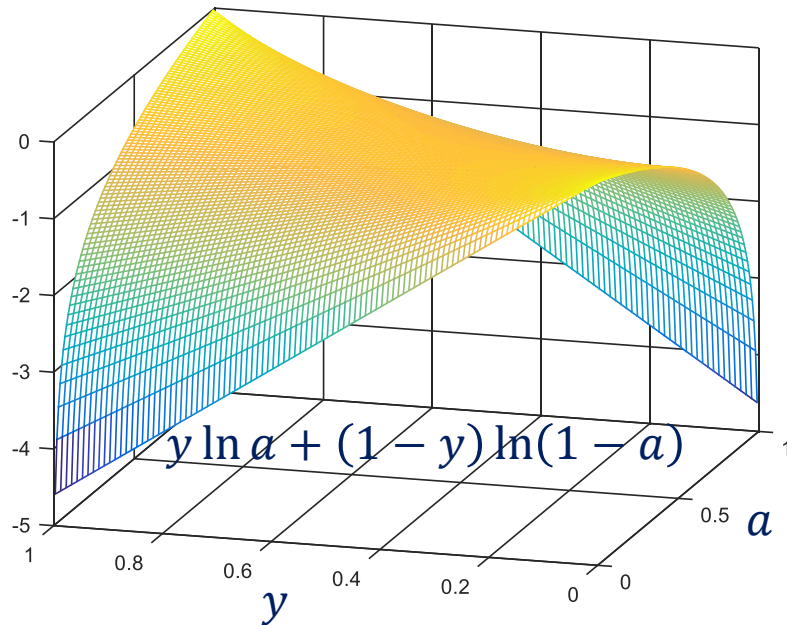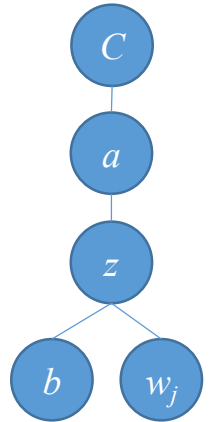$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z)$$

- ✳ sigmoid
- ＋ derivative of sigmoid
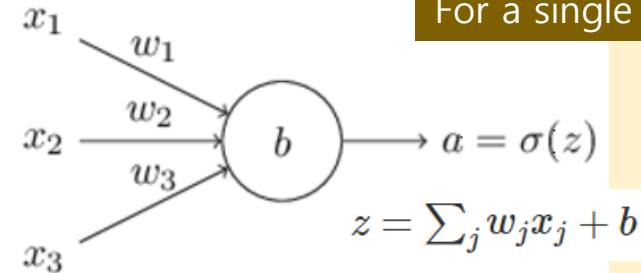
w=+0.6
b=+0.9

w=+2
b=+2

Cross Entropy Cost Function

Introducing the cross-entropy cost function

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

$y \ln a + (1-y) \ln(1-a)$

[Two features]
- Non-negative → cost function
- Zero at y=a

For a single neuron

$$z = \sum_j w_j x_j + b$$

$$a = \sigma(z)$$

Error gradient

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

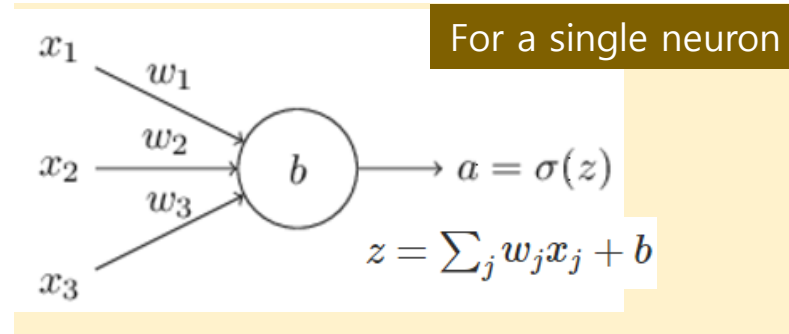$$\frac{\partial C}{\partial a} = \frac{y}{a} + (1-y)\frac{-1}{1-a} = \frac{y-a}{(1-a)a}$$

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

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

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

Cross Entropy vs MSE

Error gradient comparison

For a single neuron

$$a = \sigma(z)$$

$$z = \sum_j w_j x_j + b$$

Cross − entropy cost	*Quadratic cost*
$C = -\dfrac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$	$C = \dfrac{1}{n}\sum_x \dfrac{1}{2}(a-y)^2$
$\dfrac{\partial C}{\partial w_j} = \dfrac{1}{n}\sum_x (\sigma(z)-y)x_j$	$\dfrac{\partial C}{\partial w_j} = \dfrac{1}{n}\sum_x (\sigma(z)-y)\cdot \boxed{\sigma'(z)} \cdot x_j$
$\dfrac{\partial C}{\partial b} = \dfrac{1}{n}\sum_x (\sigma(z)-y)$	$\dfrac{\partial C}{\partial b} = \dfrac{1}{n}\sum_x (\sigma(z)-y)\cdot \boxed{\sigma'(z)}$

Error signal decreaser!!!