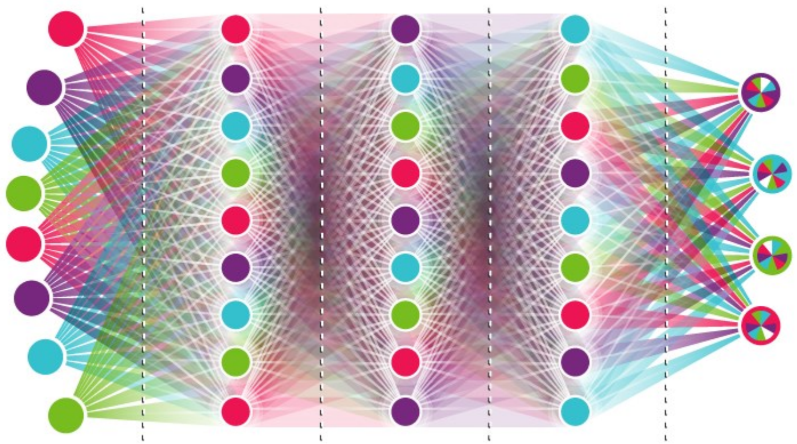
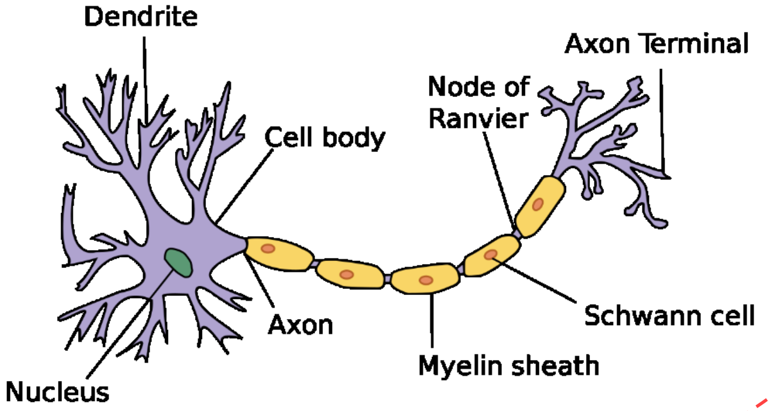
Multi-Layer Perceptron

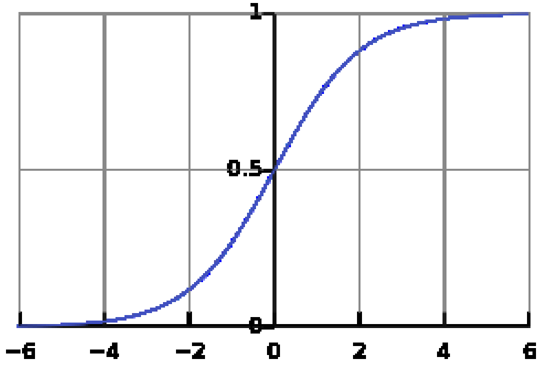
The Road to Deep Learning



Fast Campus
Start Deep Learning with TensorFlow

Recap - Perceptron





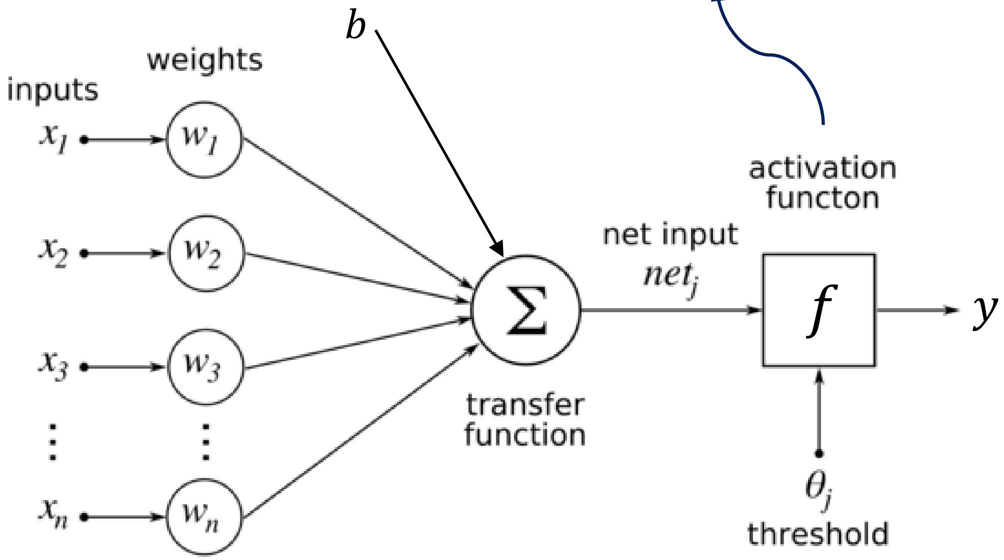
sigmoid activation function

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

$$y = f(\mathbf{wx} + \mathrm{b})$$

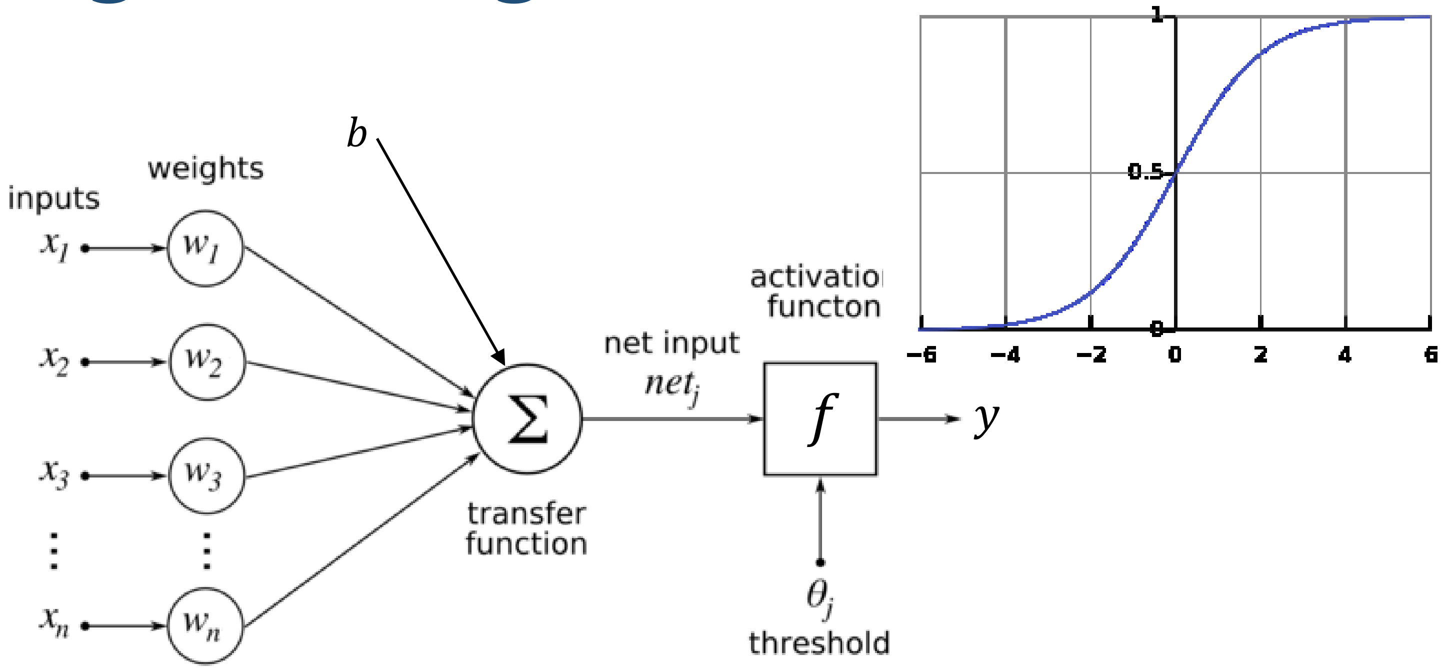
$$\boldsymbol{w} = [w_1\ w_2\ w_3\ \dots\ w_n]$$

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



<Perceptron>

Recap – Logistic Regression



<Perceptron>

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

sigmoid activation

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

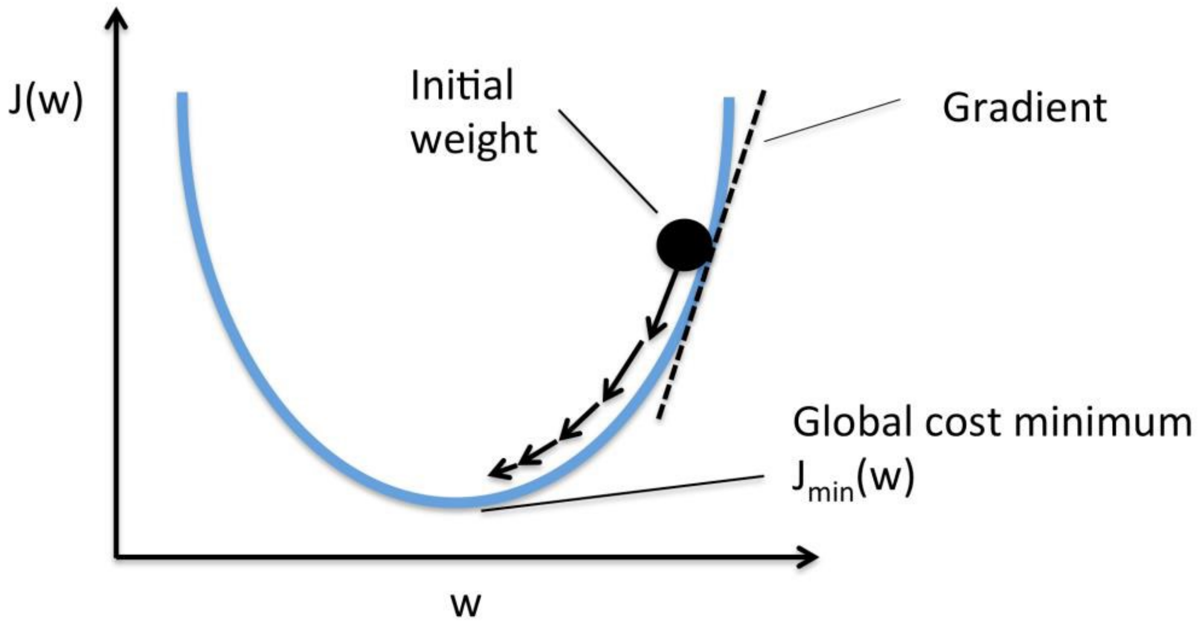
Sigmoid activation : small changes in their weights and bias cause only a small change in their output

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

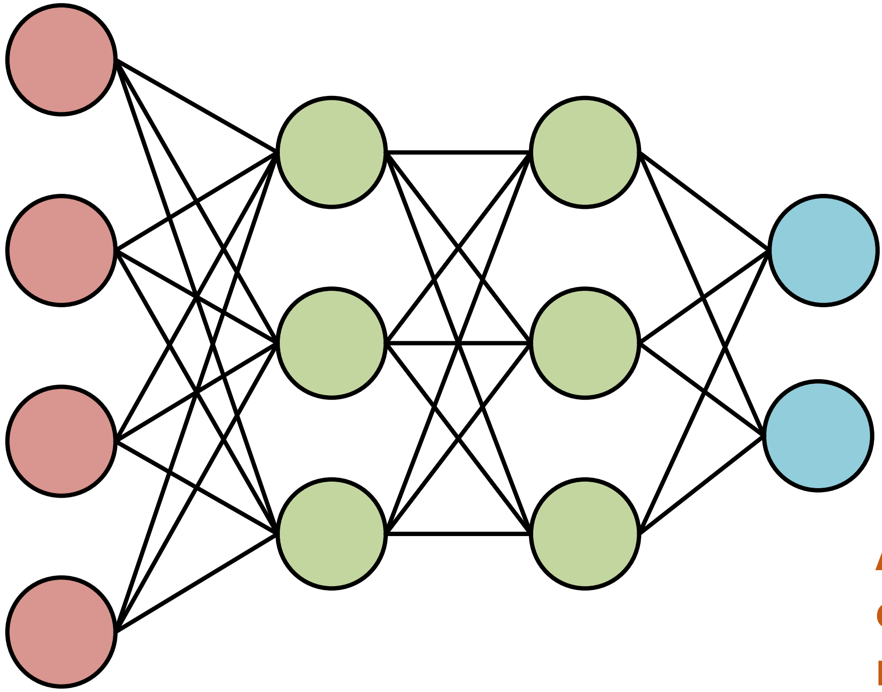
Recap – Training(Gradient Descent)

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

- 방향 : 그 지점에서의 – gradient
- 속력(보폭) : learning rate(α)



How about This? 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-Layer부터 생각해봅시다

미분을 계산해봅시다!

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

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

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

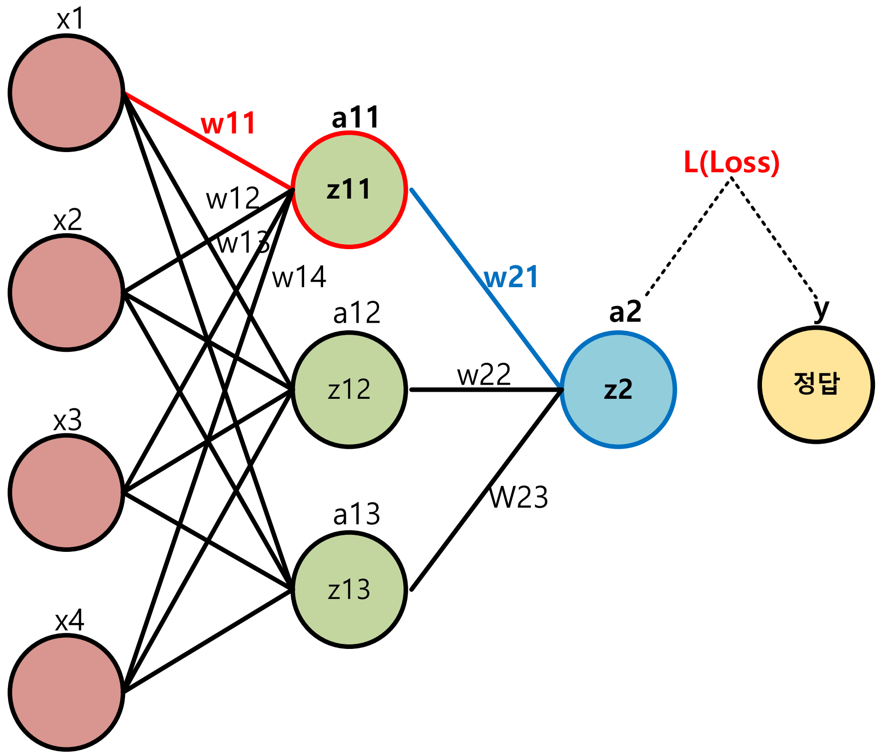
$$a2 = z2$$

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

일 때,

w11을 update 하기 위해 필요한 미분값

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



Back Propagation

$$z11 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + x4 \cdot w14$$
$$a11 = \sigma(z11) = \frac{1}{1 + e^{-z11}}$$
$$z2 = a11 \cdot w21 + a12 \cdot w22 + a13 \cdot w23$$
$$a2 = z2$$
$$= (y - a2)^2$$

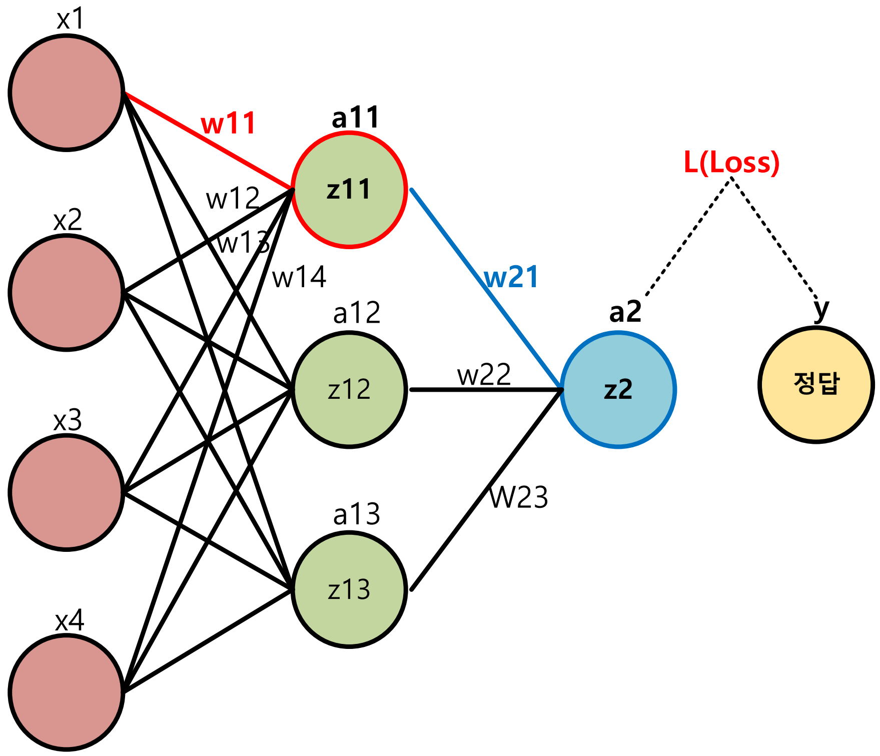
Loss부터 거꾸로 한 단계씩 미분을 해봅시다

$$\partial L / \partial a2 = -2(y - a2)$$
$$\partial a2 / \partial z2 = 1$$
$$\partial z2 / \partial a11 = w21$$
$$\partial a11 / \partial z11 = \sigma(z11) \cdot \left(1 - \sigma(z11)\right)$$
$$\partial z11 / \partial w11 = x1$$

이 미분들을 전부 각각 곱하면(chain rule),

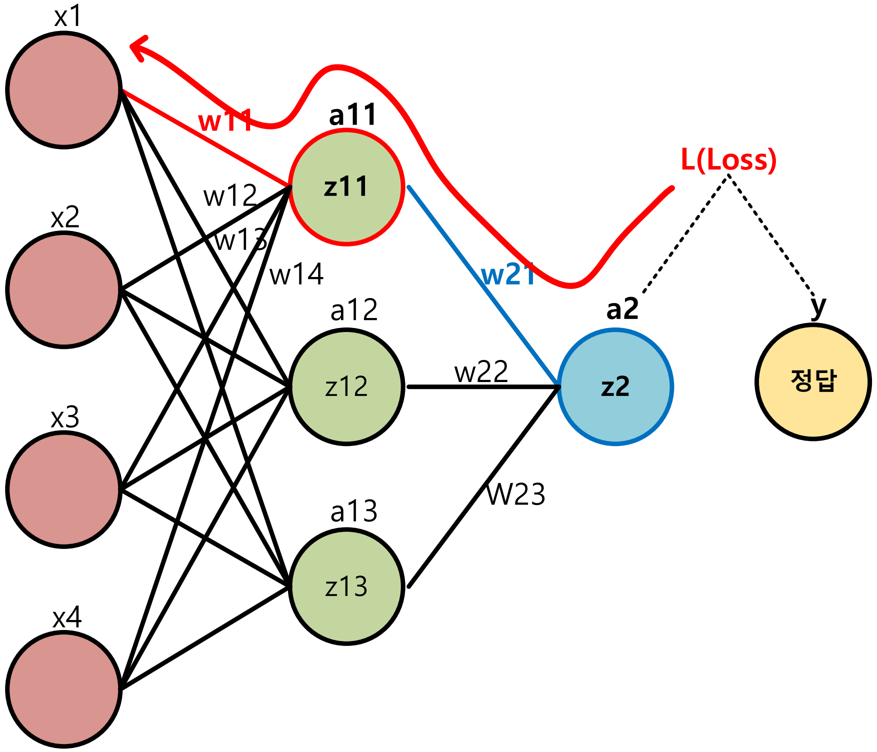
$$\frac{\partial L}{\partial a2} \cdot \frac{\partial a2}{\partial z2} \cdot \frac{\partial z2}{\partial a11} \cdot \frac{\partial a11}{\partial z11} \cdot \frac{\partial z11}{\partial w11} = \boldsymbol{\frac{\partial L}{\partial w11}}$$

우리가 구하려고 했던 미분값



Back Propagation

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

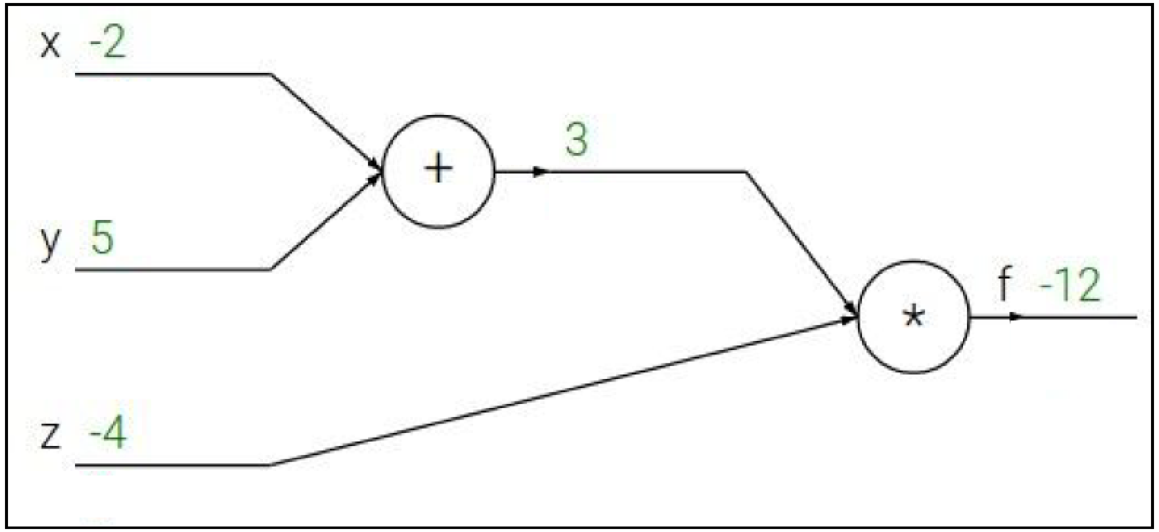


Back Propagation

Backpropagation: a simple example

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

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

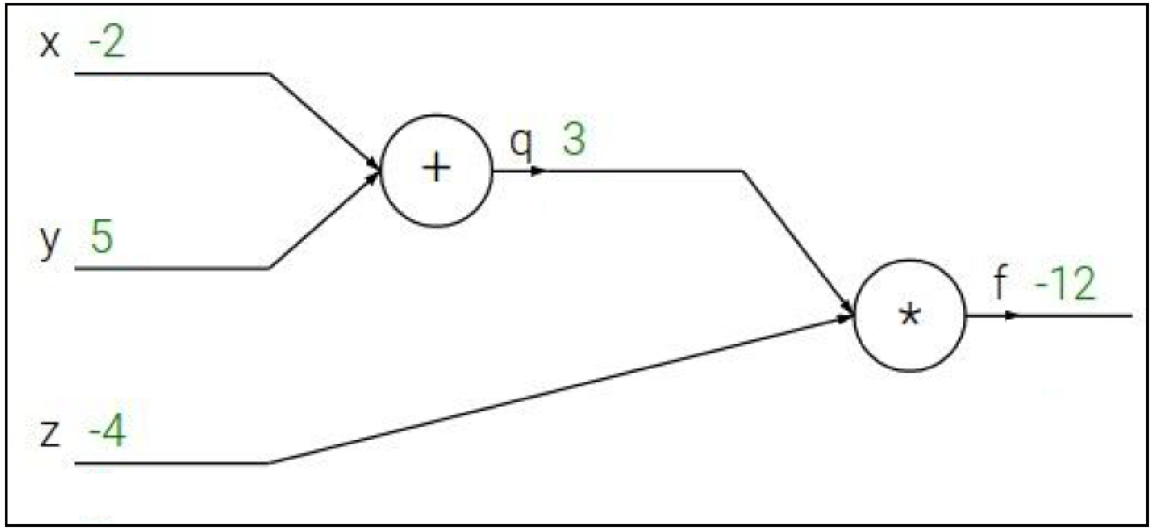


Back Propagation

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

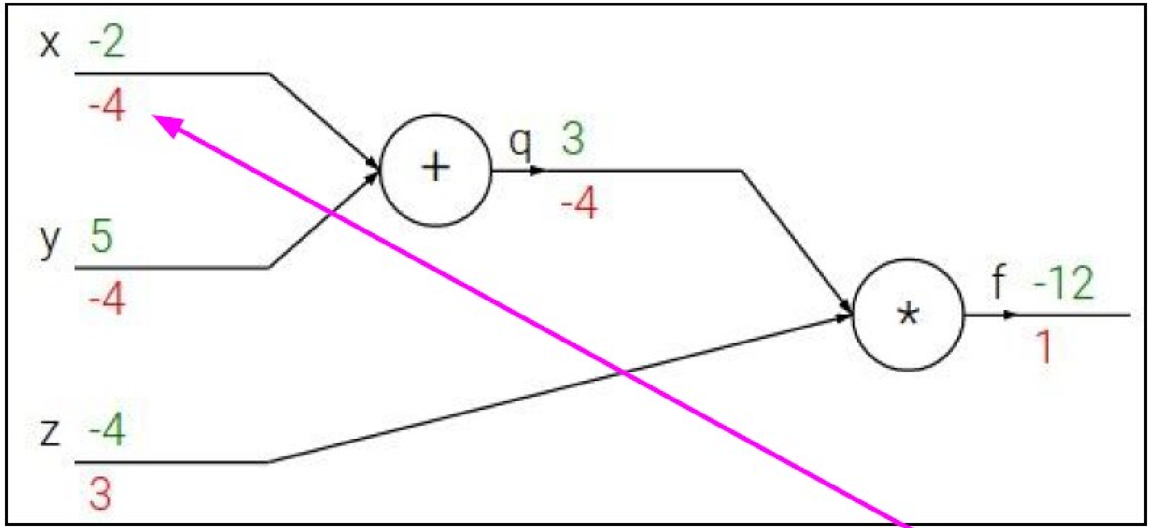
$$f(x, y, z) = (x + y)z$$

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

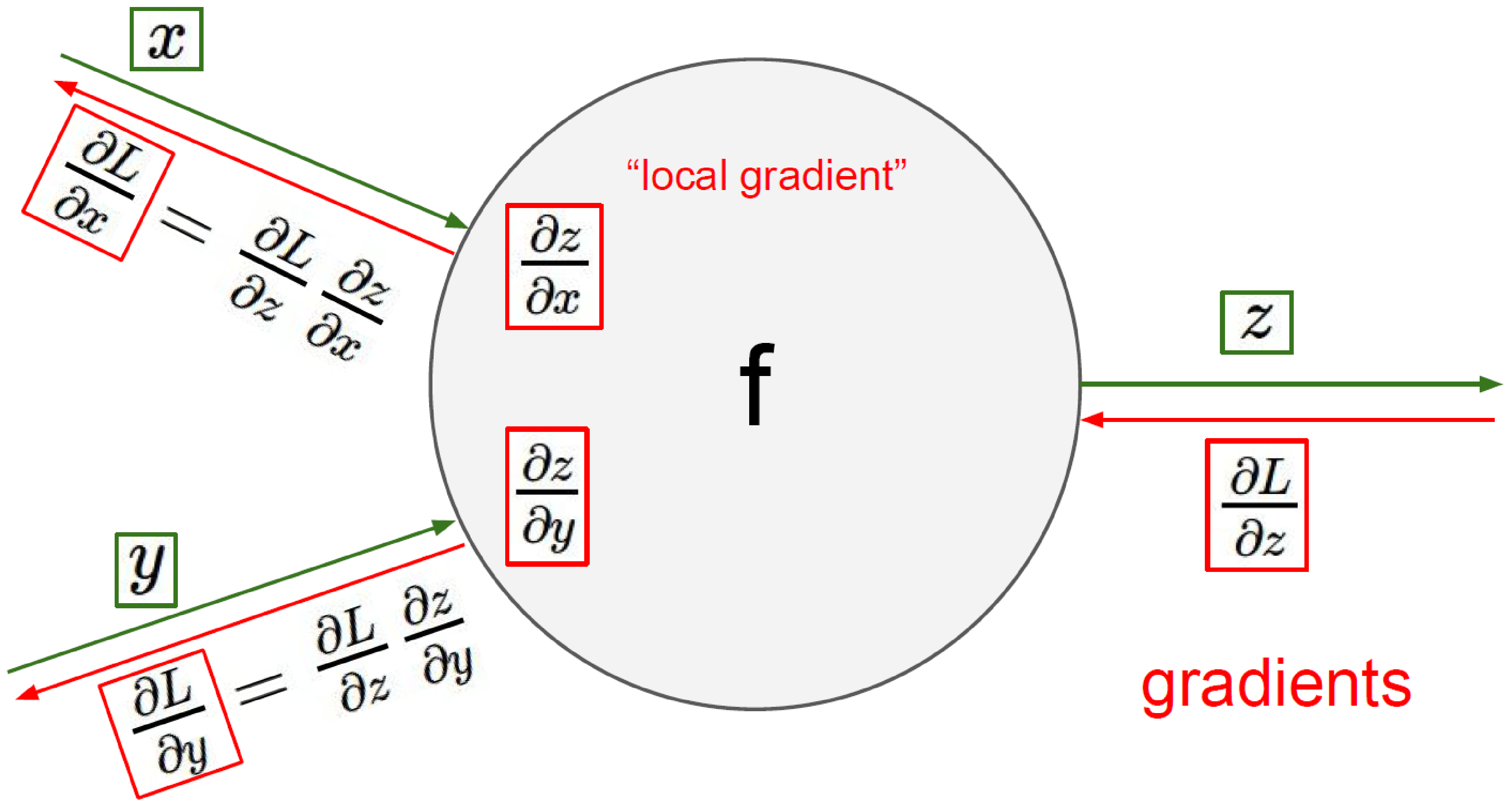


$$\frac{\partial f}{\partial x}$$

Chain rule:

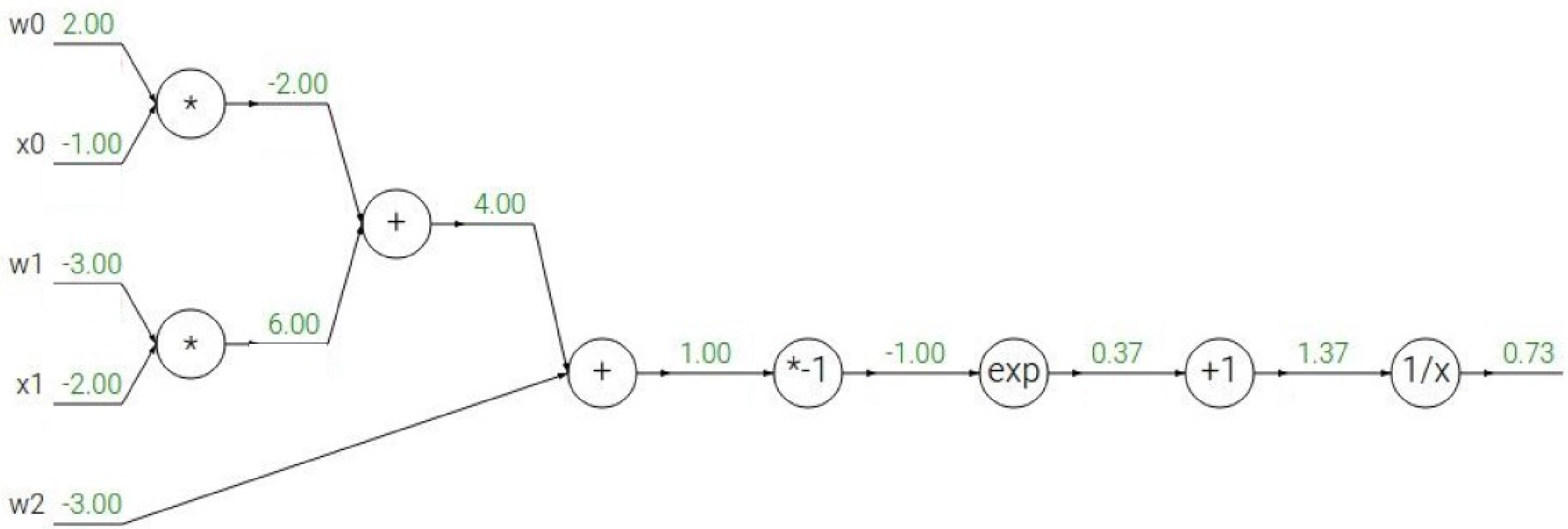
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Chain Rule(Local Gradient)



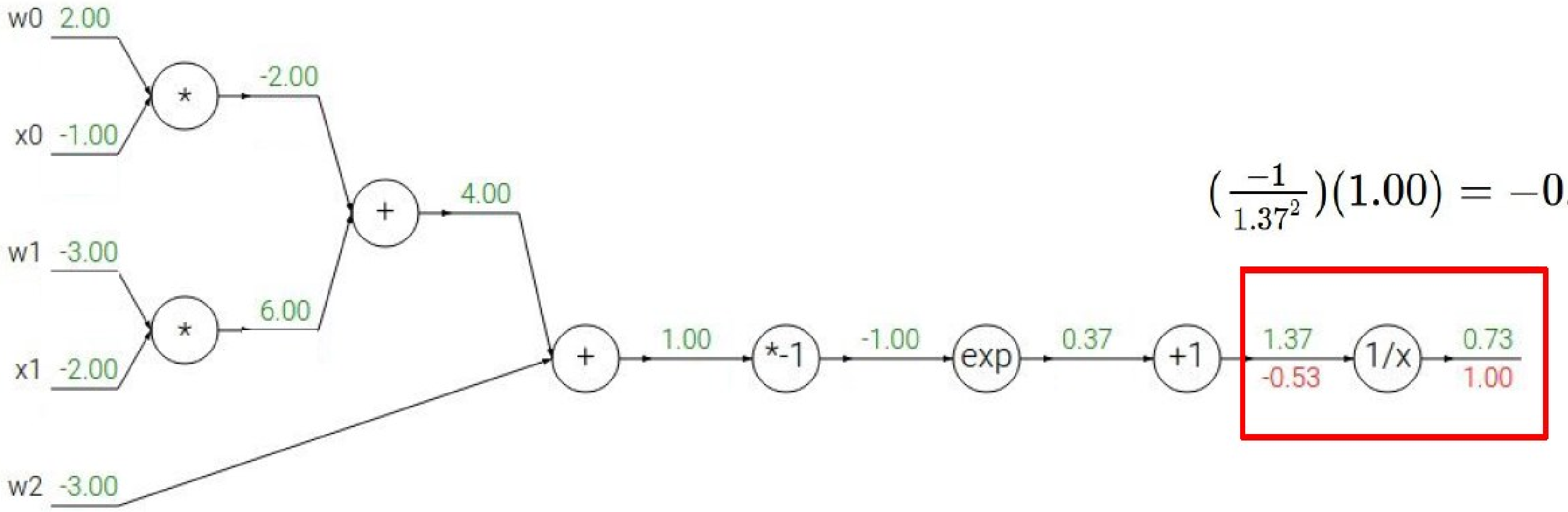
Back Propagation(Example)

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



Back Propagation(Example)

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



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

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

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

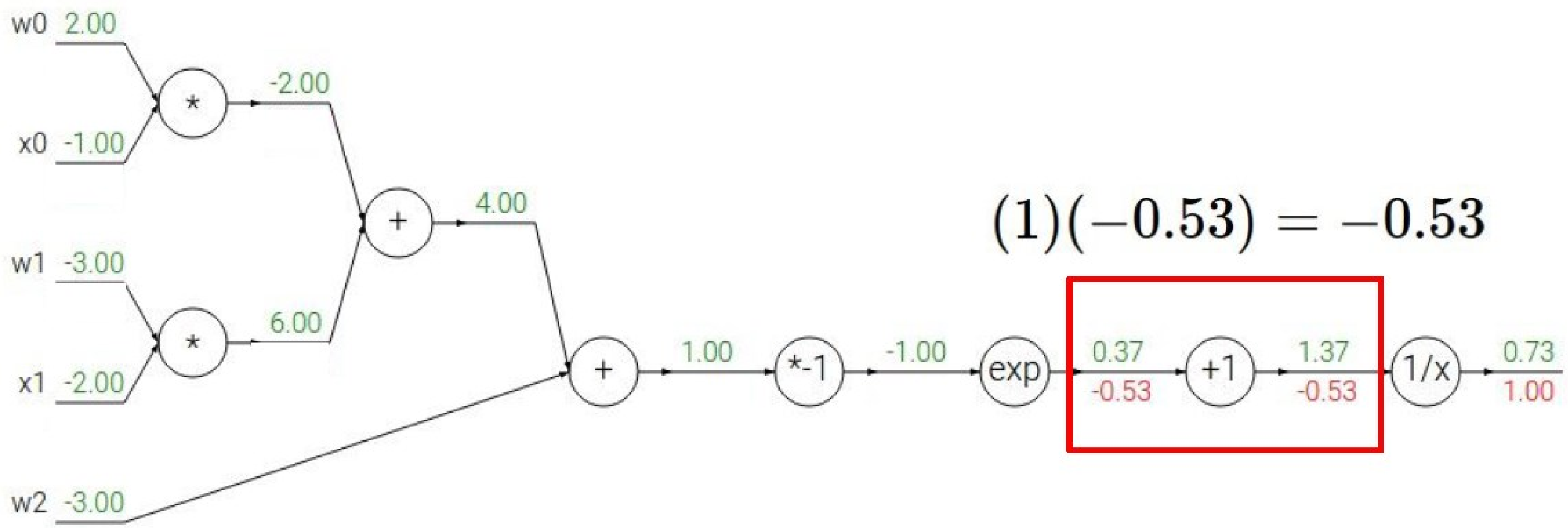
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

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

Back Propagation(Example)

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

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



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

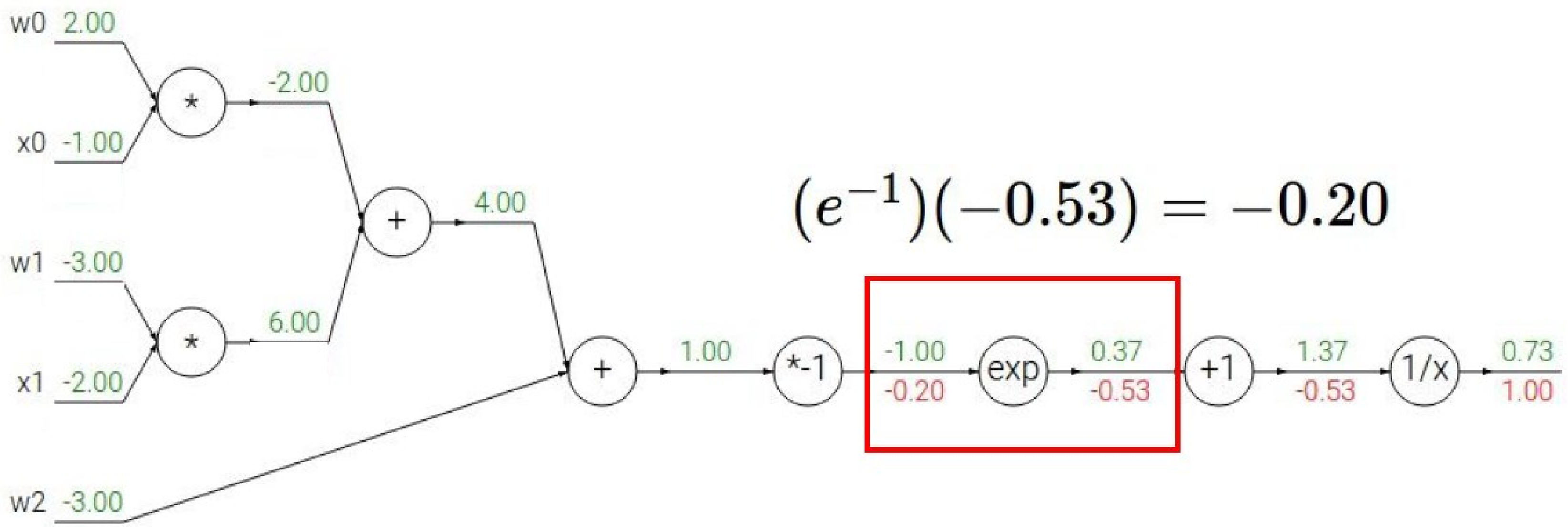
$f_a(x) = ax \quad \rightarrow \quad \dfrac{df}{dx} = a$

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

$\boxed{f_c(x) = c + x \quad \rightarrow \quad \dfrac{df}{dx} = 1}$

Back Propagation(Example)

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



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

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

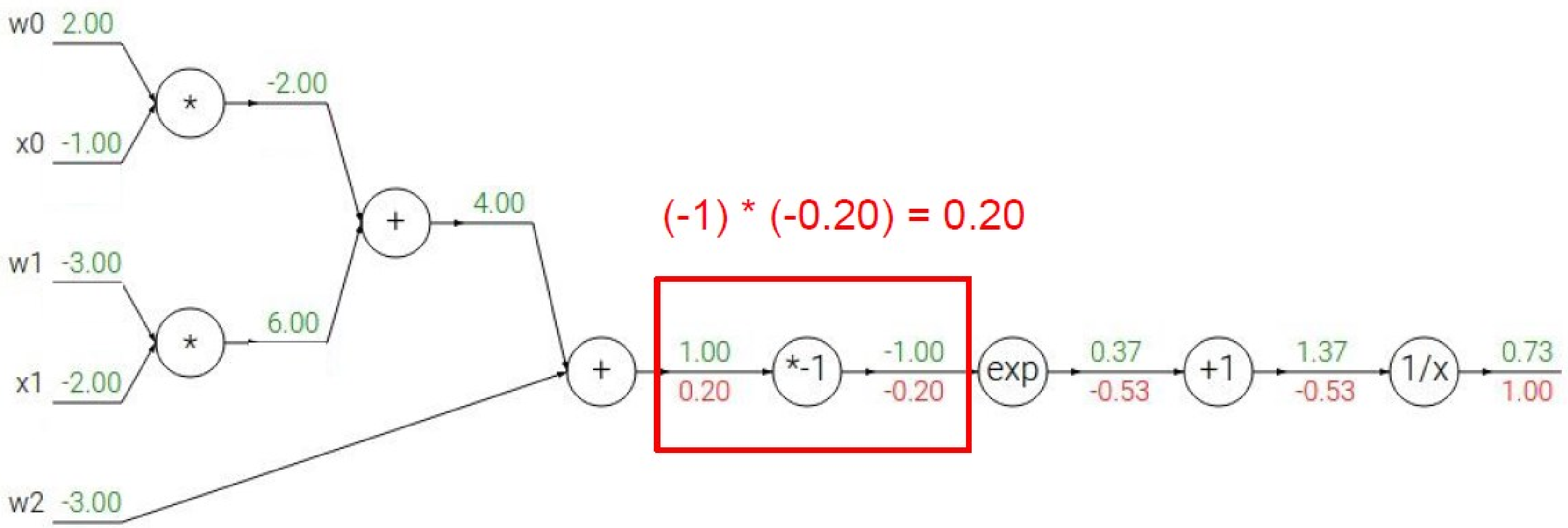
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Back Propagation(Example)

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



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

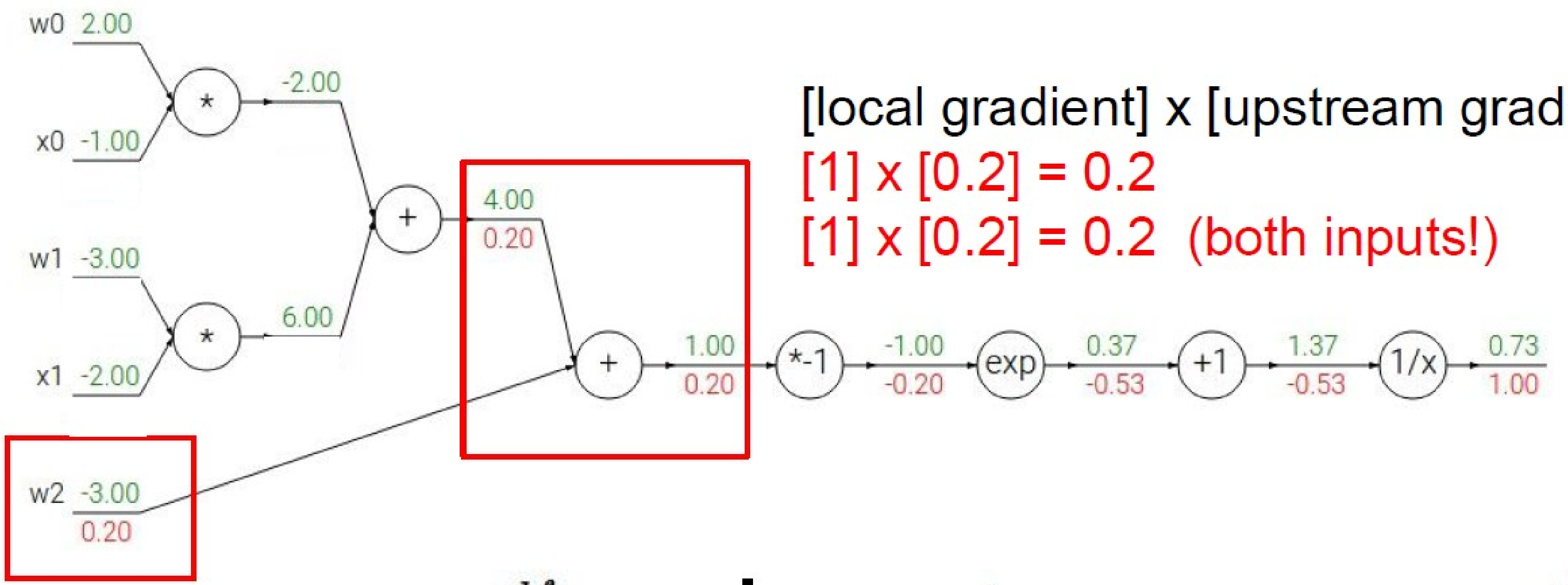
$$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$$

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

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

Back Propagation(Example)

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



[local gradient] x [upstream gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

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

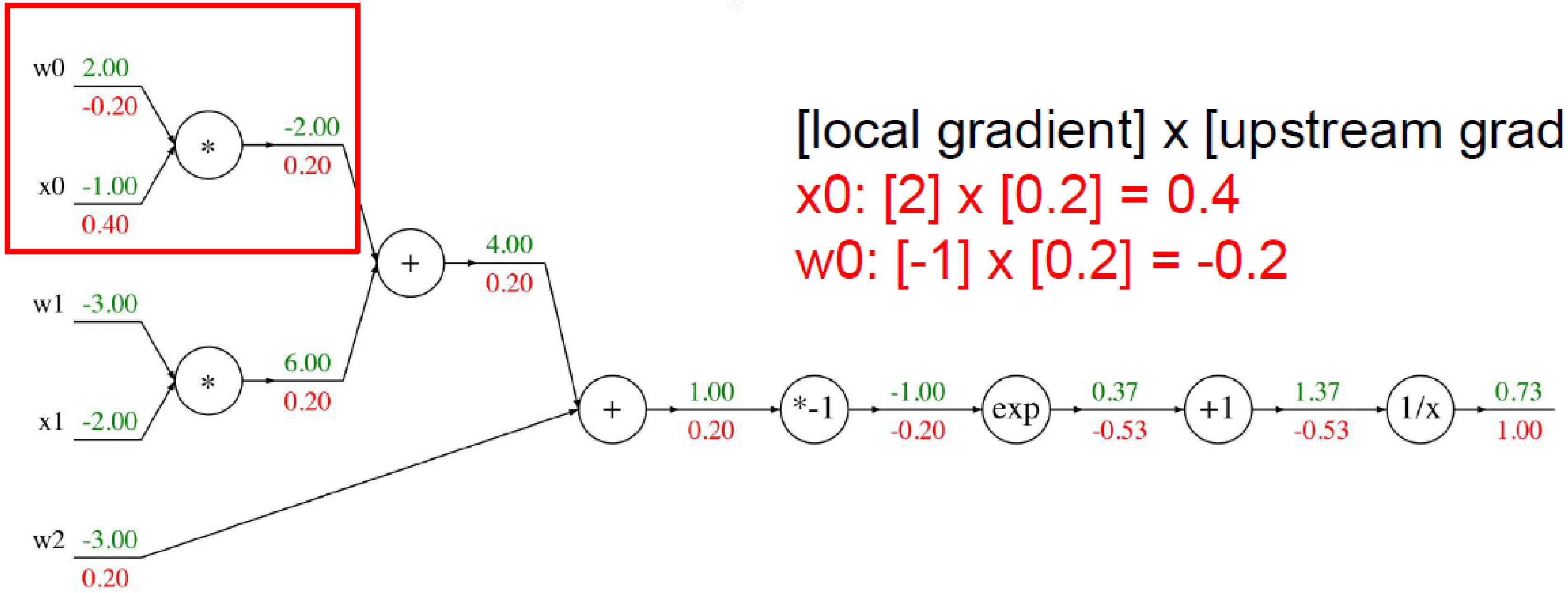
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

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

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

Back Propagation(Example)

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



[local gradient] x [upstream gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

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

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

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

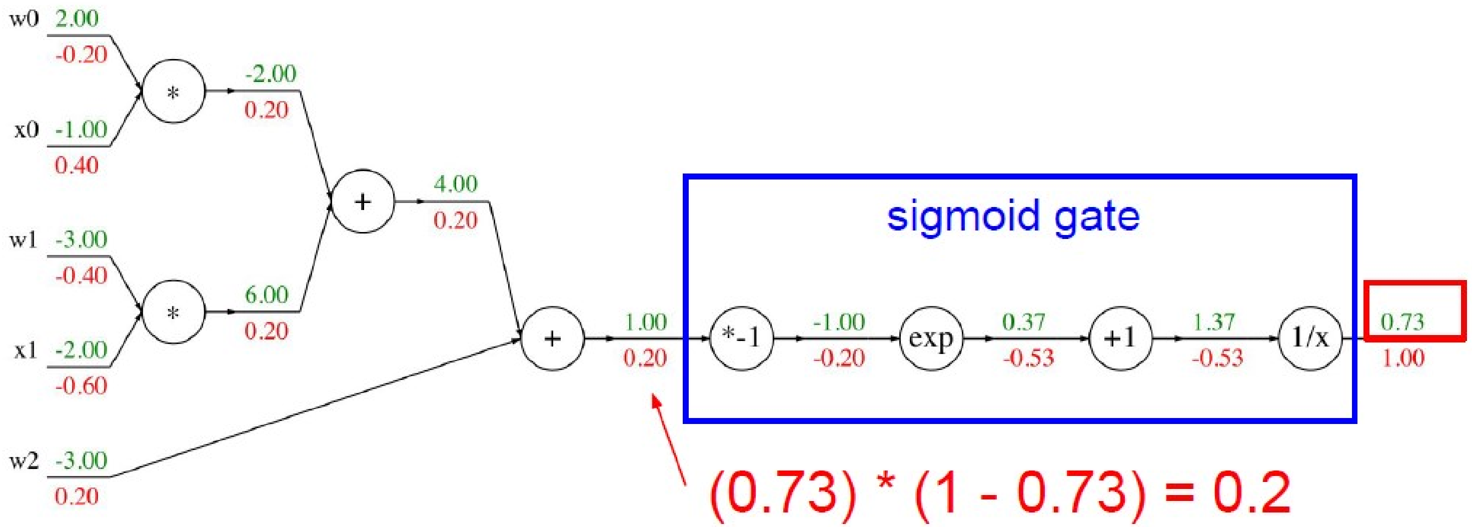
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Back Propagation(Example)

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

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

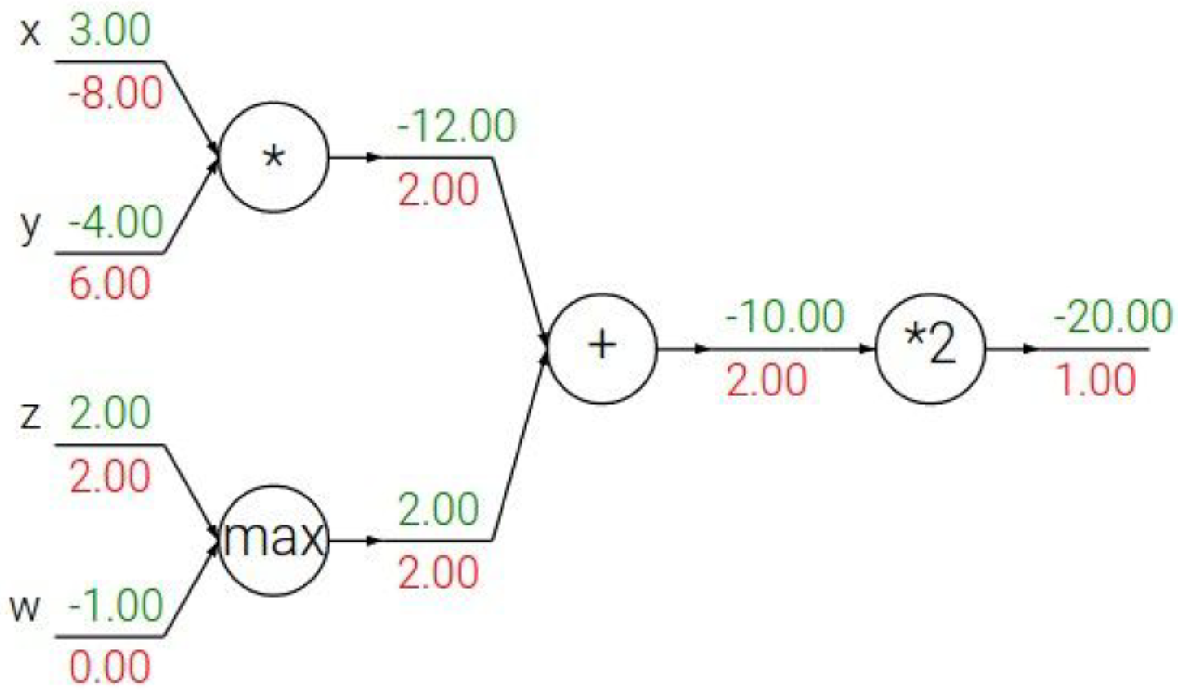
Back Propagation(Example)

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

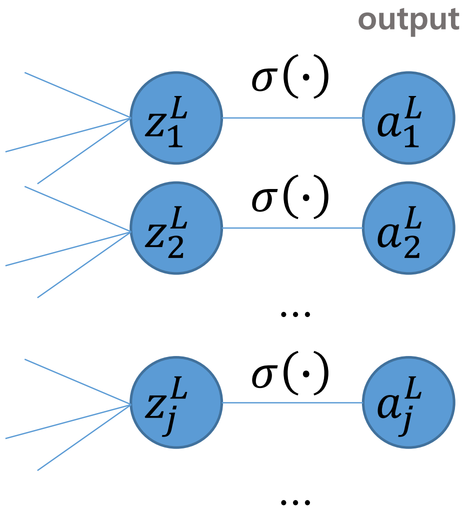
mul gate: gradient switcher



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

Softmax!

Original Output Layer



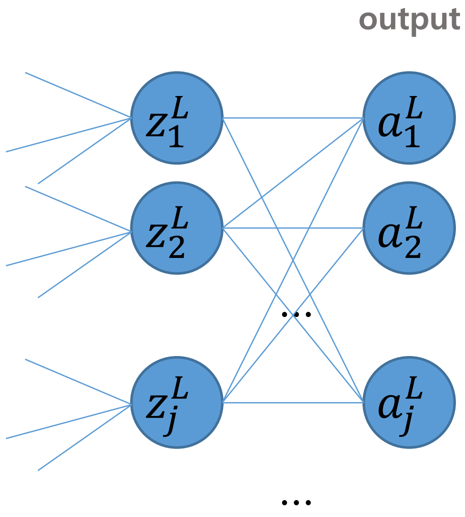
$$a_j^L = \sigma(z_j^L) = \frac{1}{1 + e^{-z_j^L}} \in [0,1]$$

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

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



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

Non-locality

$$\sum_j a_j^L = 1$$

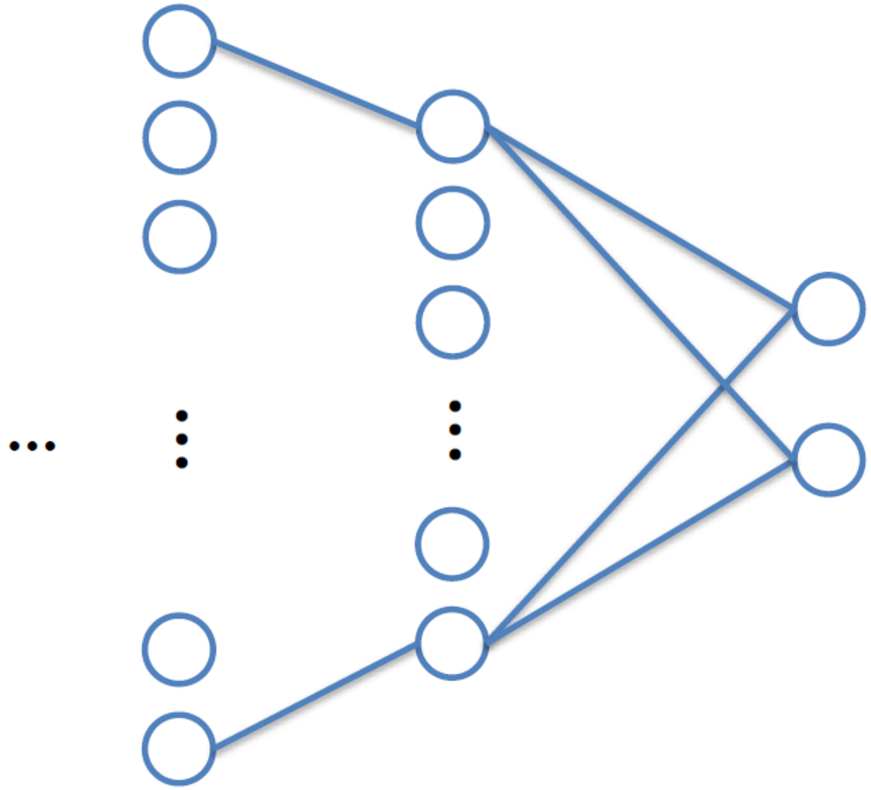
A softmax layer outputs a probability distribution!!

Monotonicity $\dfrac{\partial a_j^L}{\partial z_k^L}$ = _positive_ if j=k, _negative_ otherwise

Softmax

- Output을 확률처럼 나타내보자

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



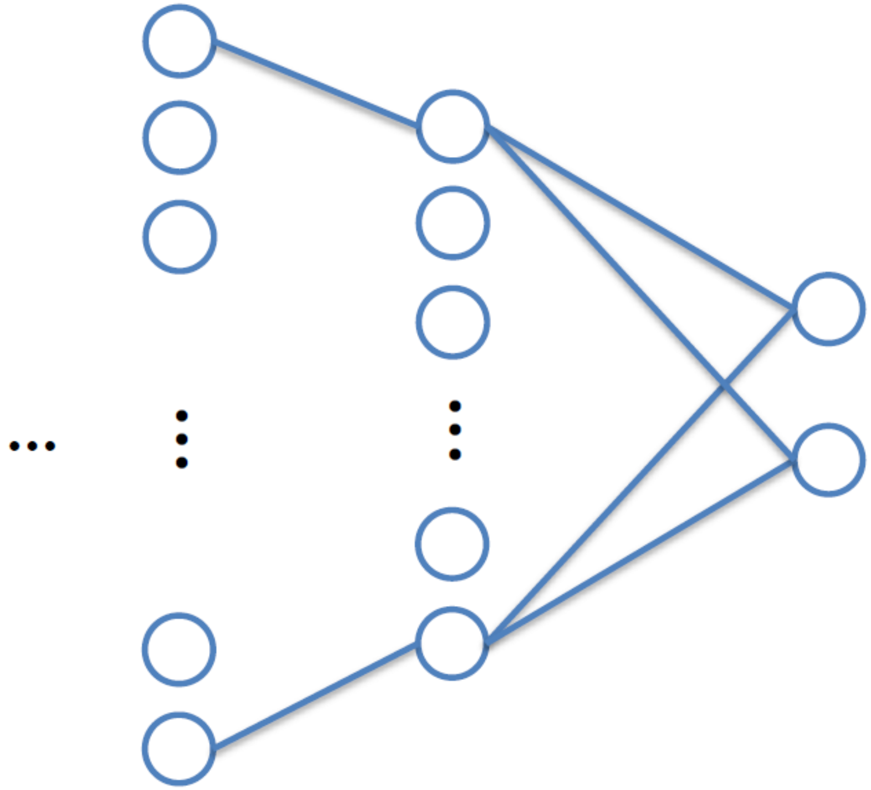
	$\mathbf{x}^\mathsf{T}\mathbf{w}_k$	$e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}$	$\frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$	one-hot vector
강아지	0.37	1.4477	0.2296	0
고양이	1.58	4.8549	0.7704	1

수치 → exp → 확률 ↔ 확률

softmax!

Loss Function of Softmax

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



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

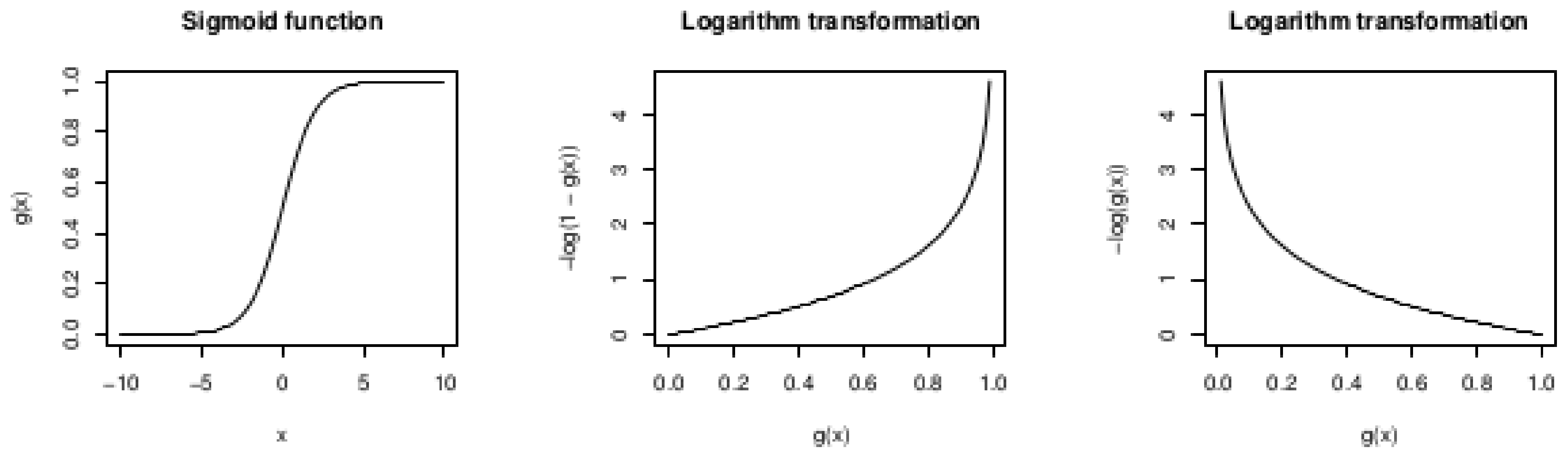
수치 → exp → 확률 ↔ 확률

softmax!

Loss Function of Logistic Regression

- Cross Entropy Loss!

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

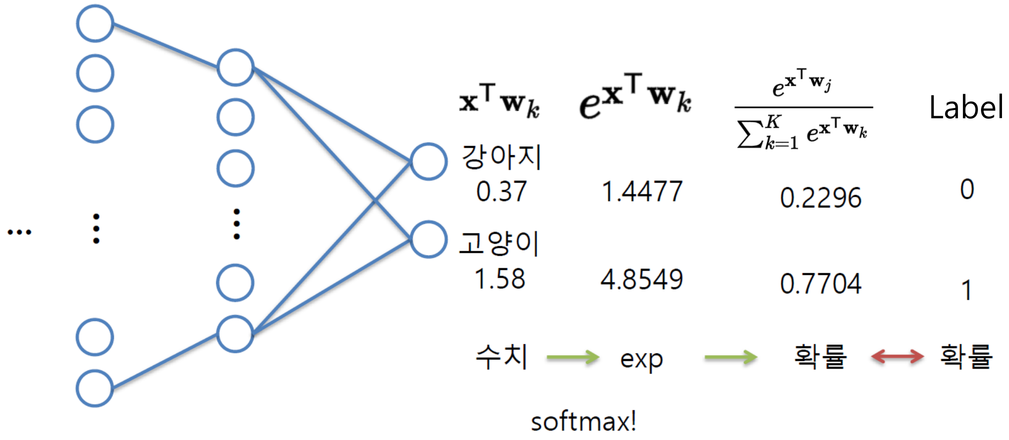


(a) *Sigmoid function.* (b) *Cost for $y = 0$.* (c) *Cost for $y = 1$.*

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

Loss Function – Classification

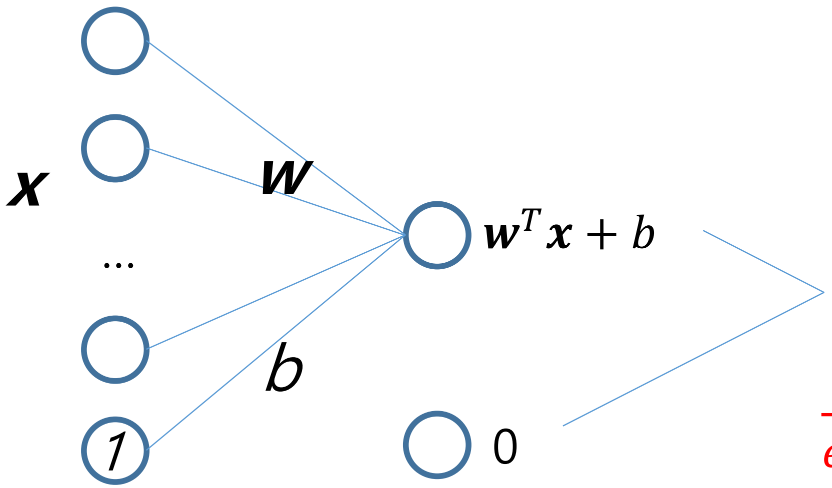
- 마지막 layer의 activation function
  - Softmax $P(y=j \mid \mathbf{x}) = \dfrac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$



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

Binary Classification

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



Softmax → sigmoid

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

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

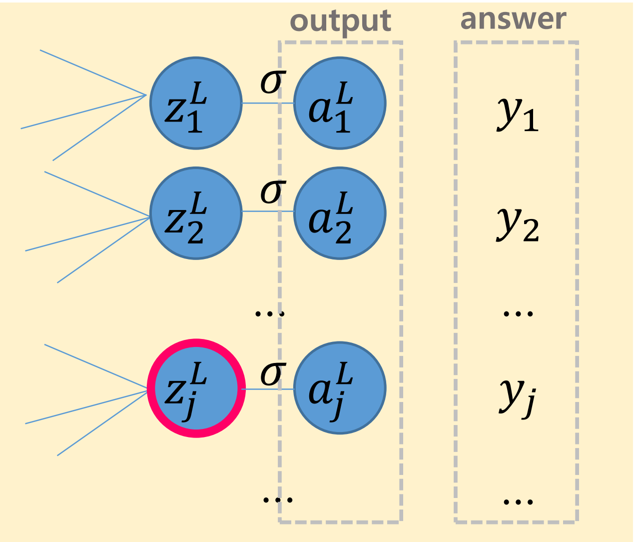
Cost	Error Signal	Error Gradient
C	$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$	$\partial C/\partial w_{jk}^l$ and $\partial C/\partial b_j^l$

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

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



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

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

Only the j[th] 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 (x) $\xrightarrow{w}$ Σ (b) $\rightarrow$ Output : 0.0 (y)

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

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

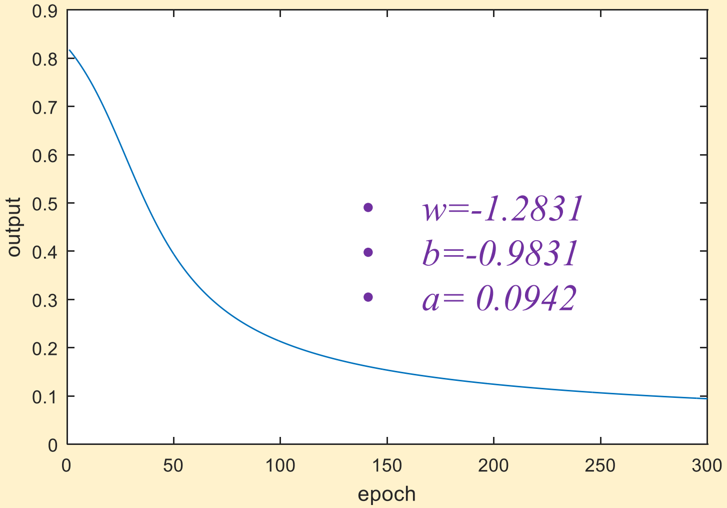
$$\delta = a\sigma'(w+b)$$

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

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

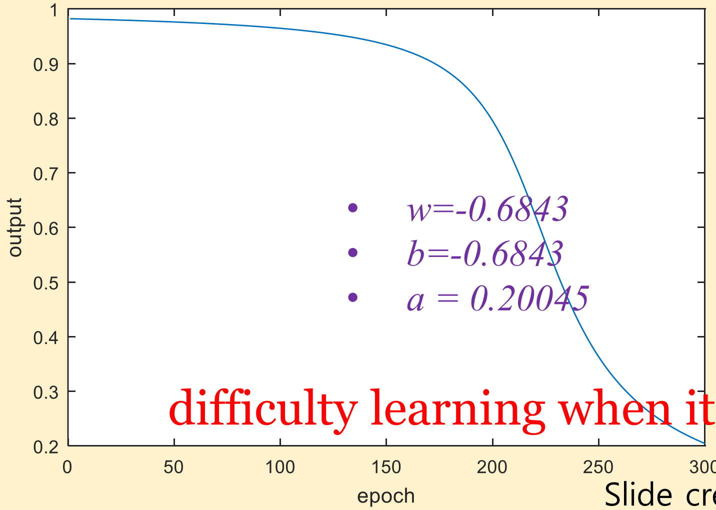
Initialization 1

Input : 1.0 (x) $\xrightarrow{w=+0.6}$ Σ ($b=+0.9$) $\rightarrow$ Output : 0.8176 (a)



Initialization 2

Input : 1.0 (x) $\xrightarrow{w=+2.0}$ Σ ($b=+2.0$) $\rightarrow$ Output : 0.9820 (a)



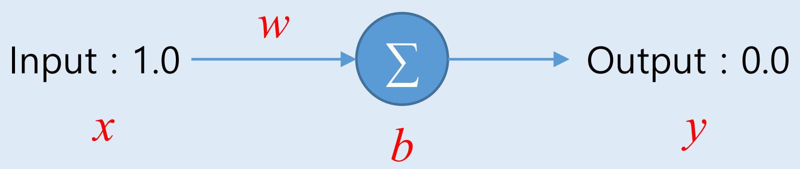
difficulty learning when it's badly wrong

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



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

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

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

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

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

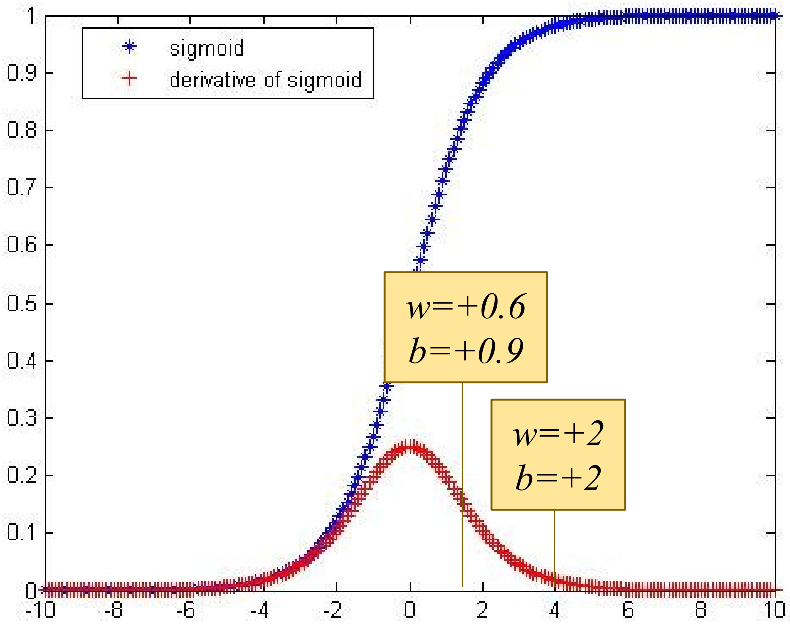
Learning slow means are $\partial C / \partial w$, $\partial C / \partial b$ small

Why they are small???

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

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

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

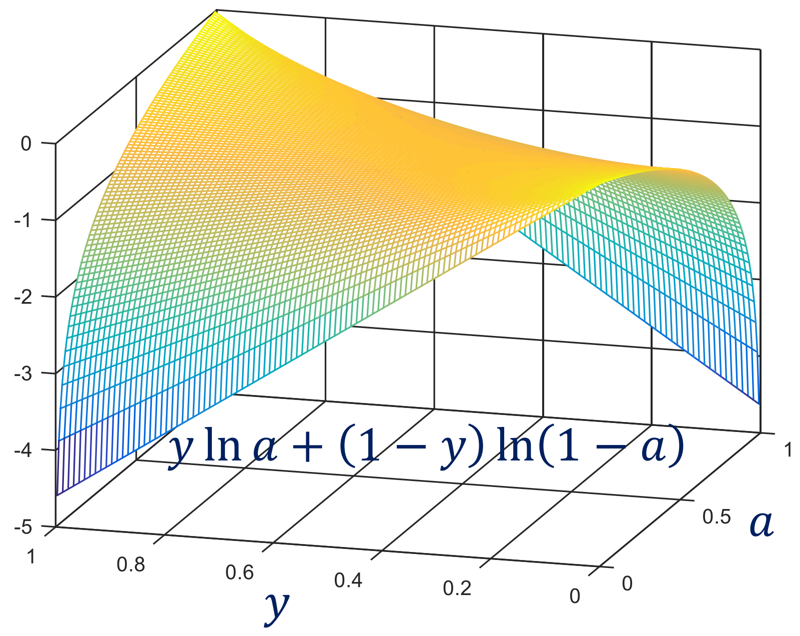


Slide credit : Hwalsuk Lee@CLAIR

Cross Entropy Cost Function

Introducing the cross-entropy cost function

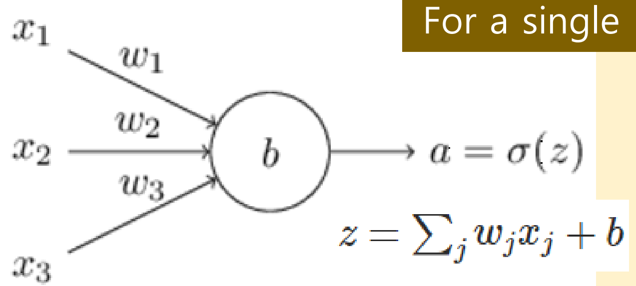
$$C = \frac{-1}{n}\sum_x [y\ ln\, a + (1-y) ln(1-a)]$$

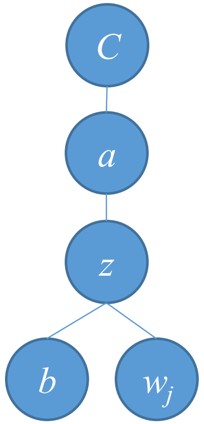


[Two features]

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

For a single neuron





Error gradient

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

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

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

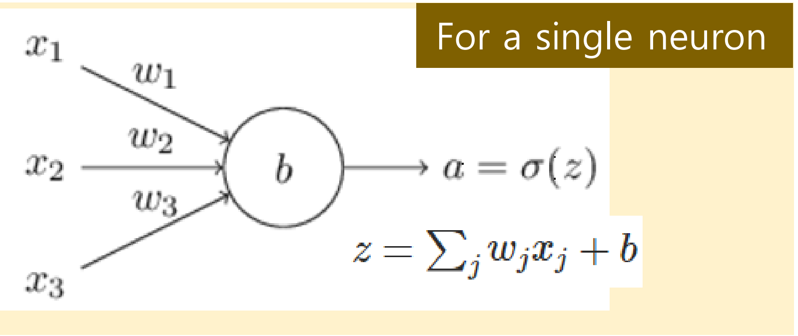
$$\frac{\partial C}{\partial w_j} = \frac{-1}{n}\sum_x (y-a)x_j = \frac{1}{n}\sum_x (\sigma(z)-y)x_j$$

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

Slide credit : Hwalsuk Lee@CLAIR

Cross Entropy vs MSE

Error gradient comparison



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

Slide credit : Hwalsuk Lee@CLAIR