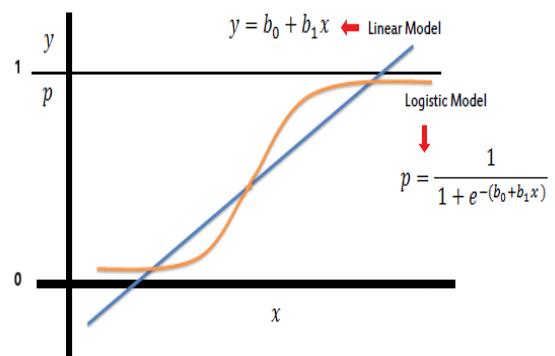
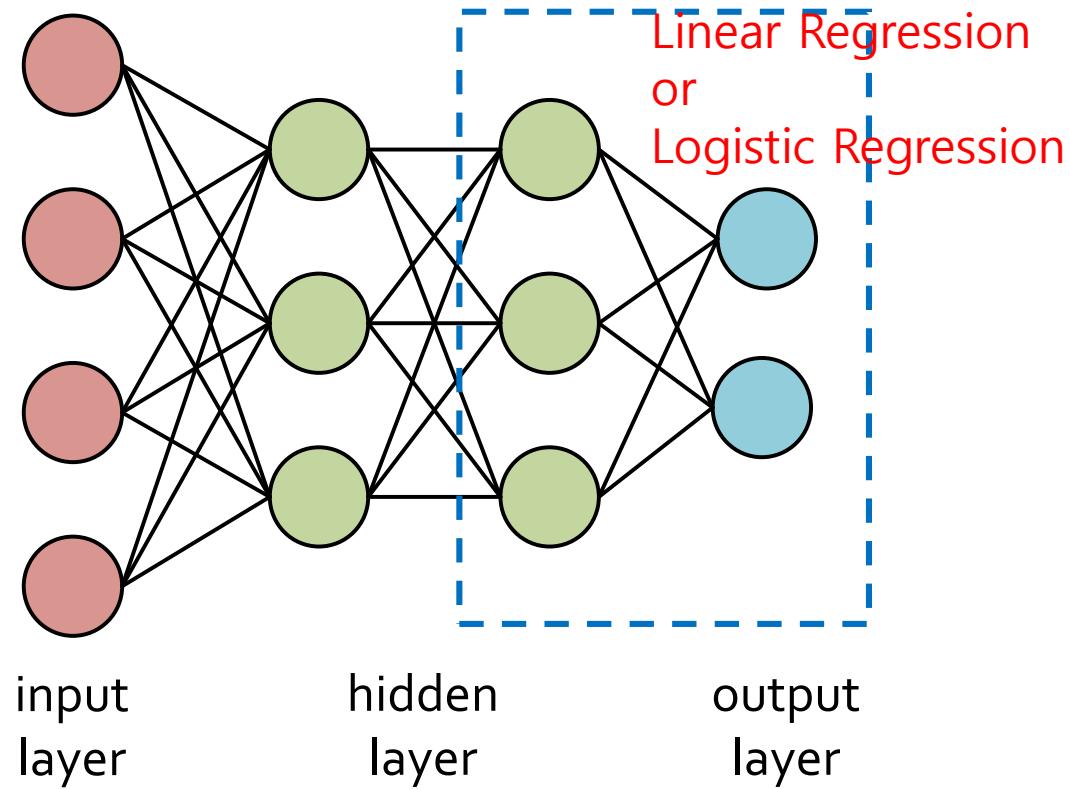


Linear Regression & Logistic Regression

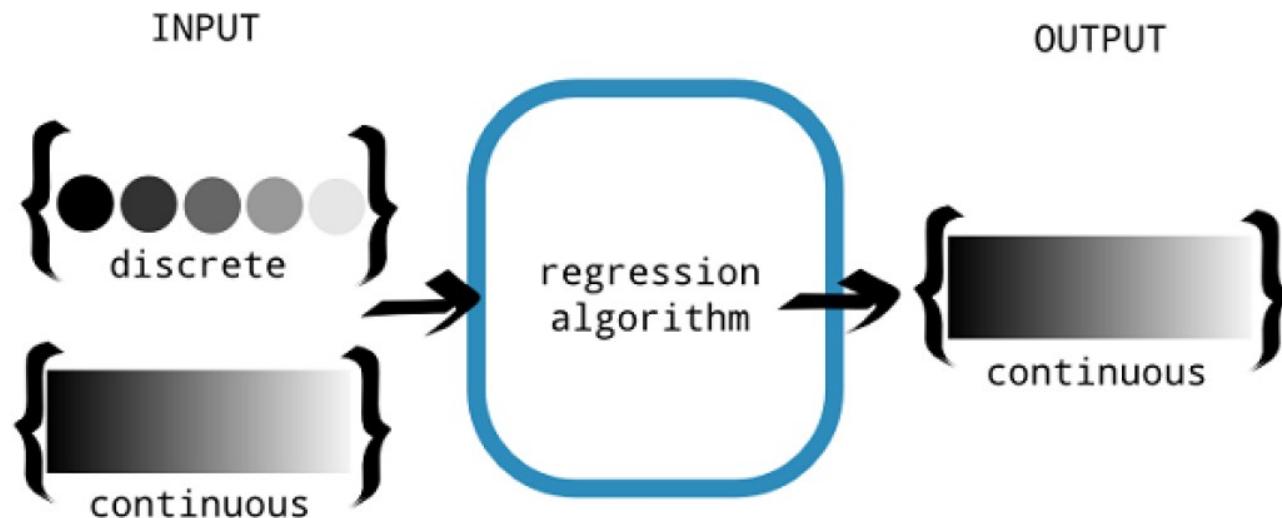


Deep Learning Uses Linear Regression/Logistic Regression



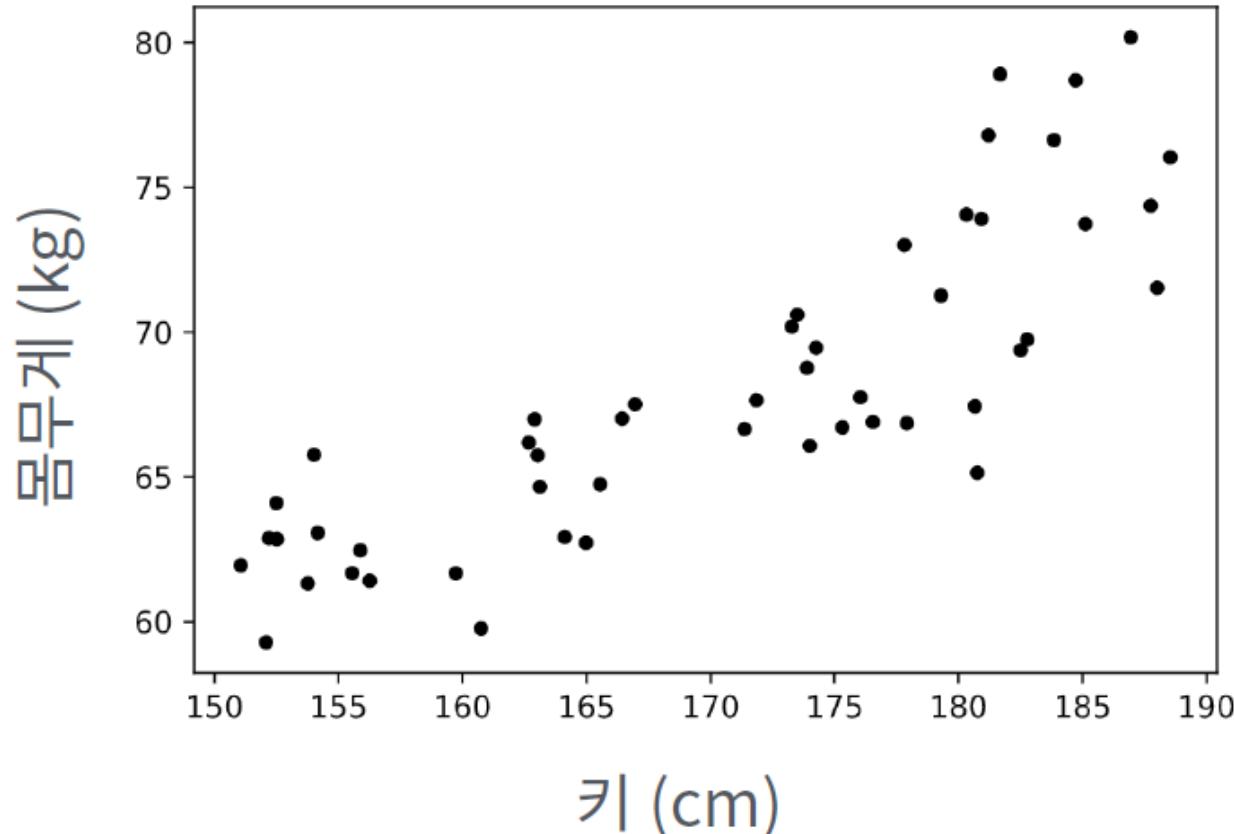
Regression

- Classification
 - Is it a dog or not? (binary)
 - What is this animal? (multi-class)
- Regression
 - What will be the stock value? $y \in \mathbb{R}$
 - What will be the housing price? $y \in [0, \infty)$



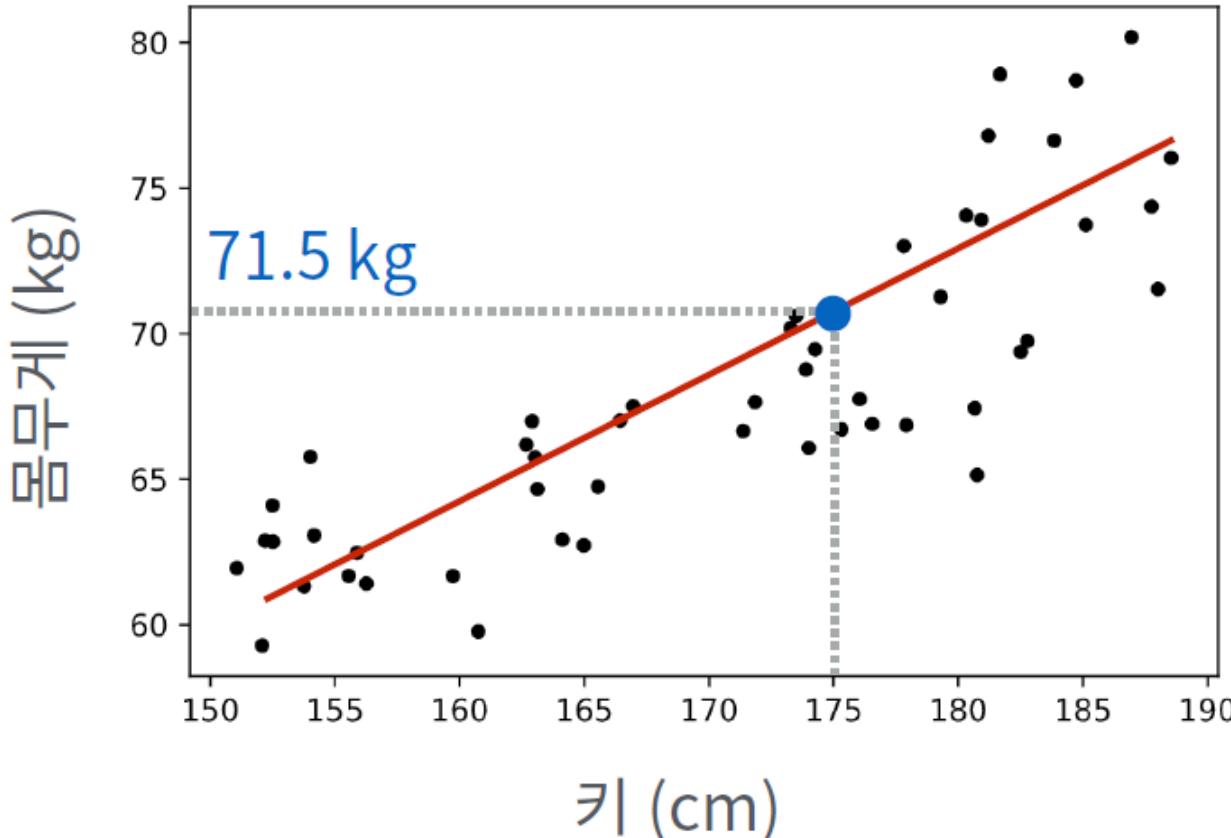
Linear Regression

- 어느 학교 학생들의 신체검사 자료
- 새로 전학온 학생 A의 키가 175cm일 때 예상 몸무게는?



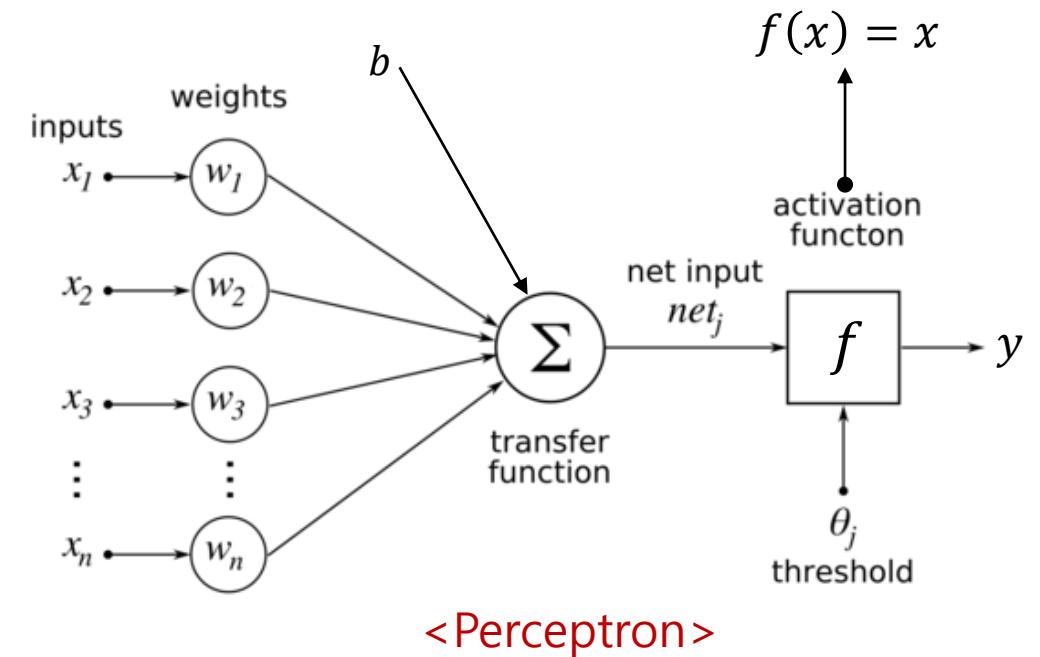
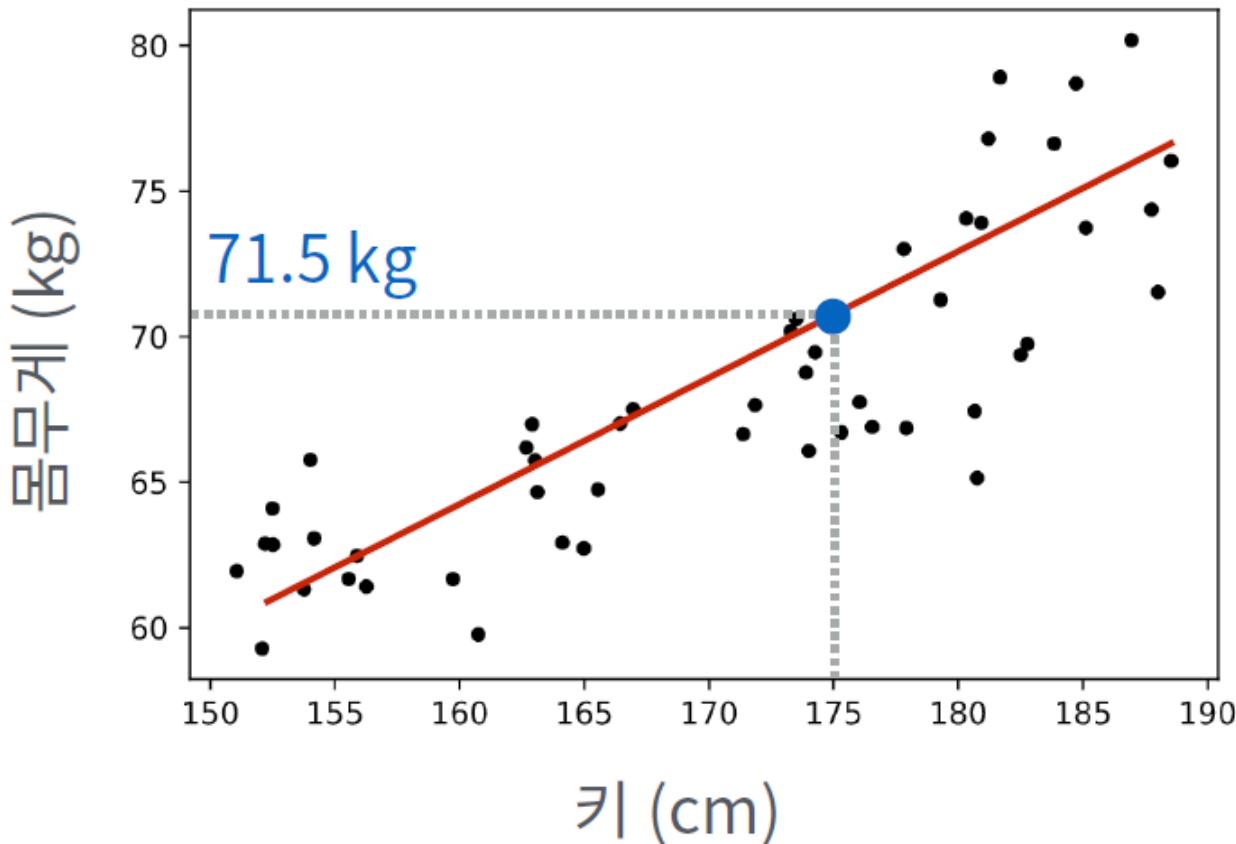
Linear Regression

- 어느 학교 학생들의 신체검사 자료
- 새로 전학온 학생 A의 키가 175cm일 때 예상 몸무게는?



Linear Regression

- 선형함수(예 : 1차함수)로 주어진 data를 근사한다
- $y = wx + b$



<Perceptron>

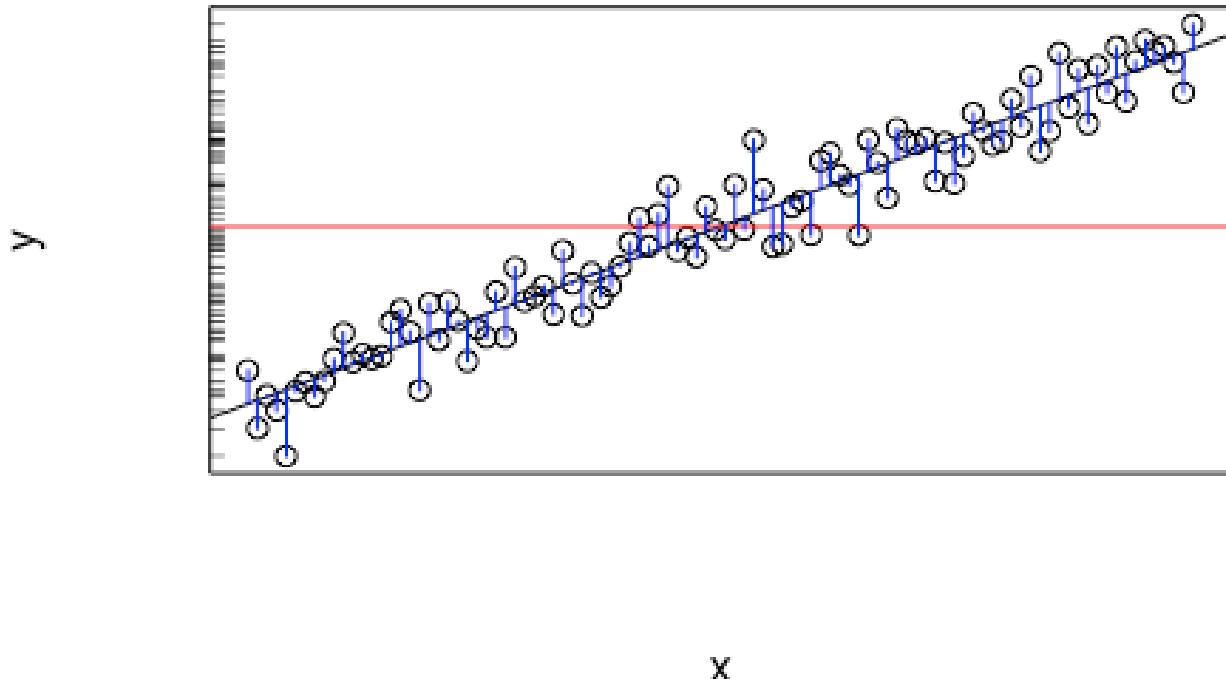
$$y = f(\mathbf{wx} + b)$$

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

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

Linear Regression

- 잘 예측했는지 측정할 척도(metric)가 필요함



$$y^* = wx + b \text{ (예측값)}$$

$$\begin{aligned} Loss &= \sum_i (y_i - y_i^*)^2 \\ &= \sum_i (y_i - wx_i - b)^2 \end{aligned}$$

Linear Regression

- Loss 값을 minimize하는 w 와 b 를 구하면 될텐데.... 어떻게?
 - Random Search – 가능????
 - Cost function을 미분해서 최솟값(미분=0이 되는 점)을 찾자!

b 구하기

$$L = \sum_i (y_i - wx_i - b)^2$$

$$\frac{\delta L}{\delta b} = \frac{\delta \sum_i (y_i - wx_i - b)^2}{\delta b}$$

$$= -2 \sum_i (y_i - wx_i - b) = ny_{avg} - nwx_{avg} - nb = 0$$

$$\therefore b = y_{avg} - wx_{avg}$$

w 구하기

$$L = \sum_i (y_i - wx_i - b)^2$$

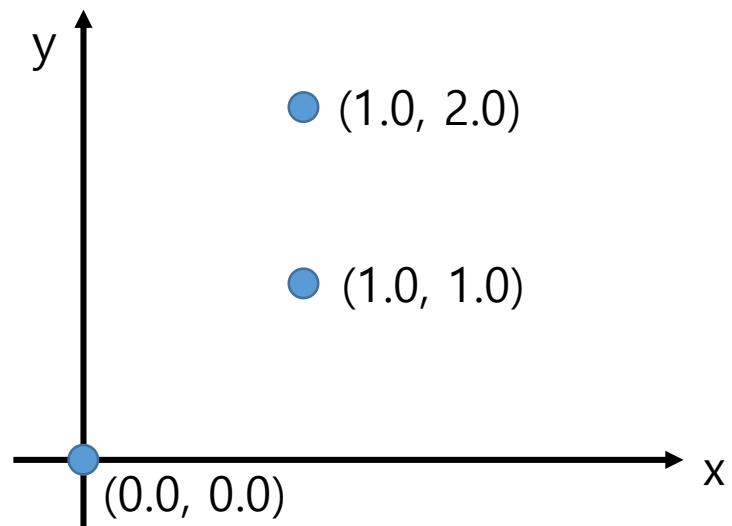
$$\frac{\delta L}{\delta w} = \frac{\delta \sum_i (y_i - wx_i - b)^2}{\delta w}$$

$$= -2 \sum_i x_i (y_i - wx_i - b) = -2 \sum_i x_i (y_i - wx_i - y_{avg} + wx_{avg})$$

$$= 0$$

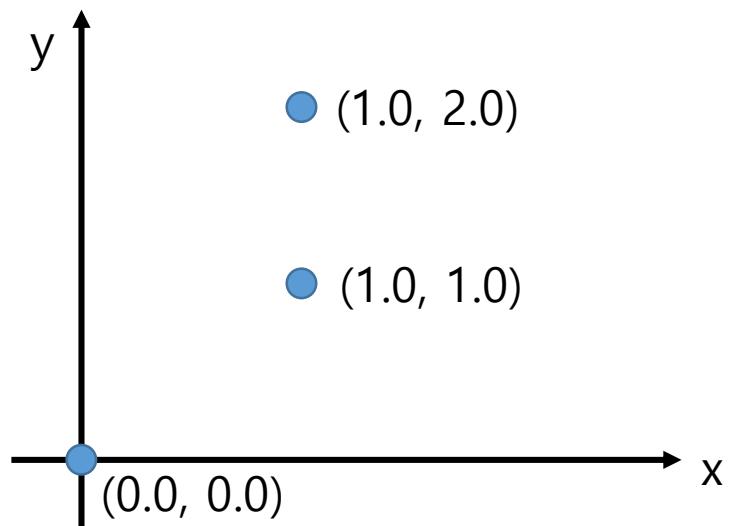
Example

- Find the linear function(f) that best describes the given data
 - $H(x, w_0, w_1) = w_1x + w_0$



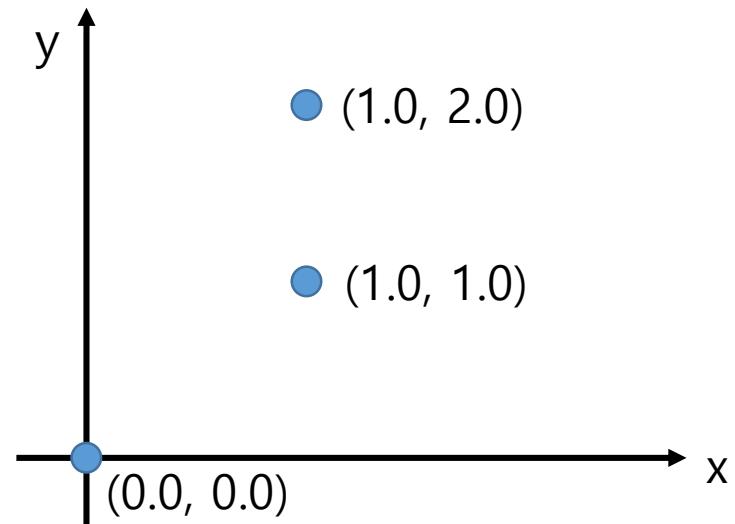
Example

- $H(0, w_0, w_1) \approx 0.0$
- $H(1, w_0, w_1) \approx 1.0$
- $H(1, w_0, w_1) \approx 2.0$



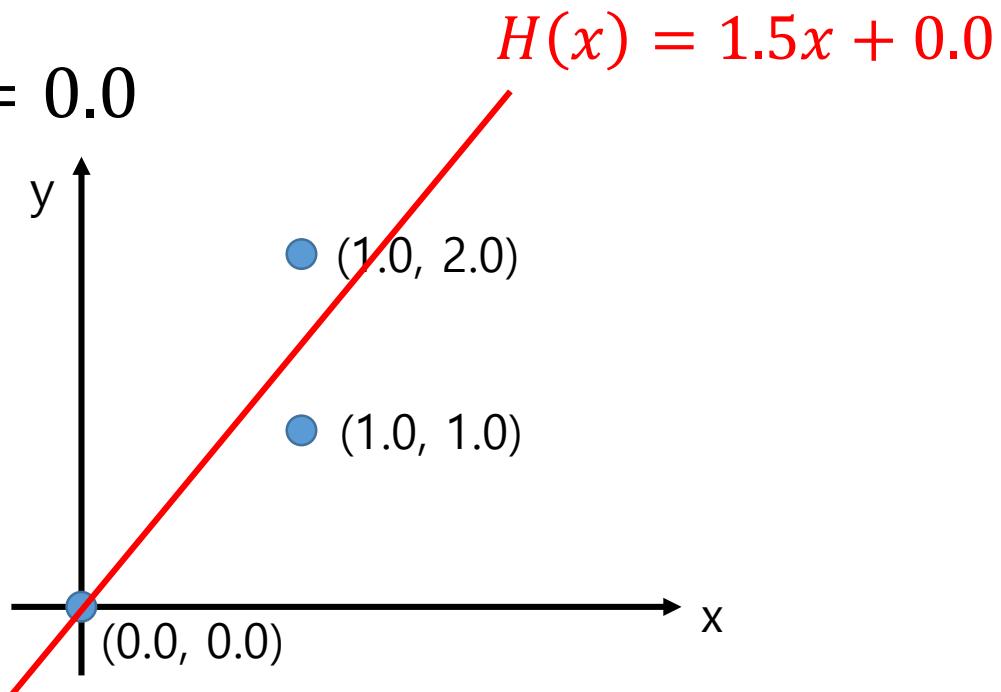
Example

- $$\begin{aligned} L &= \sum_i (y_i - w_1 x_i - w_0)^2 \\ &= (0.0 - w_1 \cdot 0.0 - w_0)^2 + (1.0 - w_1 \cdot 1.0 - w_0)^2 + (2.0 - w_1 \cdot 1.0 - w_0)^2 \\ &= 2w_1^2 + 3w_0^2 - 6w_1 - 6w_0 + 4w_1 w_0 + 5 \end{aligned}$$



Example

- $\frac{\partial L}{\partial w_1} = 4w_1 + 4w_0 - 6 = 0$
- $\frac{\partial L}{\partial w_0} = 4w_1 + 6w_0 - 6 = 0$
- $\therefore w_1 = 1.5, w_0 = 0.0$



Multi Variable Linear Regression

- x가 scalar값(1개)가 아니라 vector가 된다면??

- Input

- X_1 : Facebook 광고료
- X_2 : TV 광고료
- X_3 : 신문 광고료

- Output

- 판매량

FB	TV	신문	판매량
X_1	X_2	X_3	Y
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
:	:	:	:

Multi Variable Linear Regression

$$\begin{aligned}\mathbf{w}_{\text{lin}} &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{\text{in}}(\mathbf{w}) \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} (\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y})\end{aligned}$$

Multi Variable Linear Regression

- Find w that minimize $E_{in}(w)$ by requiring

$$\nabla E_{in}(w) = \mathbf{0}$$

- From previous equation

$$\nabla E_{in}(w) = \frac{2}{N}(X^\top X w - X^\top y)$$

$$\begin{aligned} w_{\text{lin}} &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{in}(\mathbf{w}) \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} \|X\mathbf{w} - \mathbf{y}\|^2 \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} (\mathbf{w}^\top X^\top X \mathbf{w} - 2\mathbf{w}^\top X^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}) \end{aligned}$$

Multi Variable Linear Regression

- Two scenarios
 - If $X^T X$ is invertible Moore-Penrose Pseudoinverse
 $w = \boxed{\quad} y$
 - If $X^T X$ is not invertible
Pseudo-inverse defined, but no unique solution

<Note>

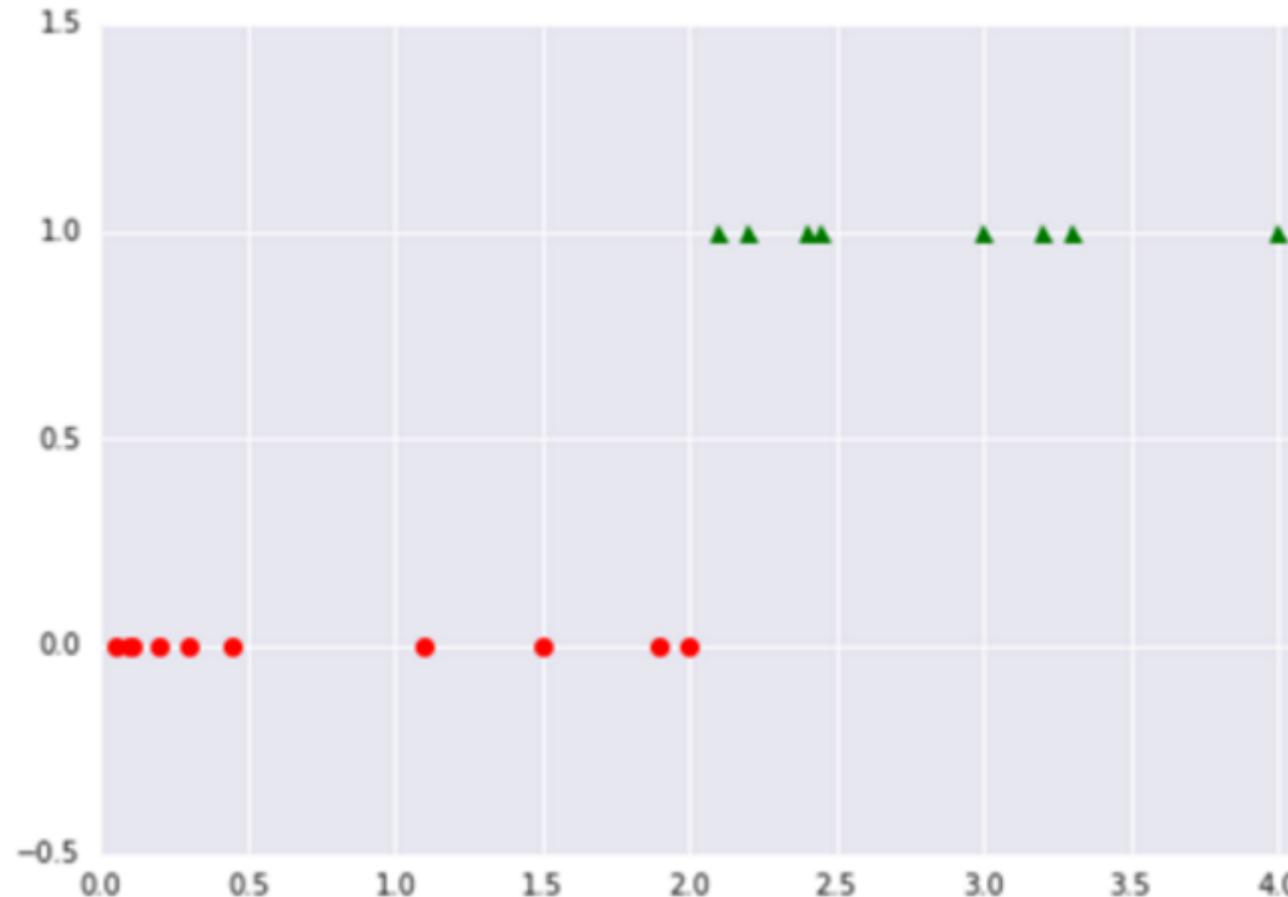
$X \in \mathbb{R}^{n \times p}$ 일 때(n: sample 수, p: input vector size),

- p가 커질수록 matrix inversion 연산량이 많아짐
- $p > n$ 이면 $X^T X$ is not invertible

Classification도 할 수 있지 않을까요?

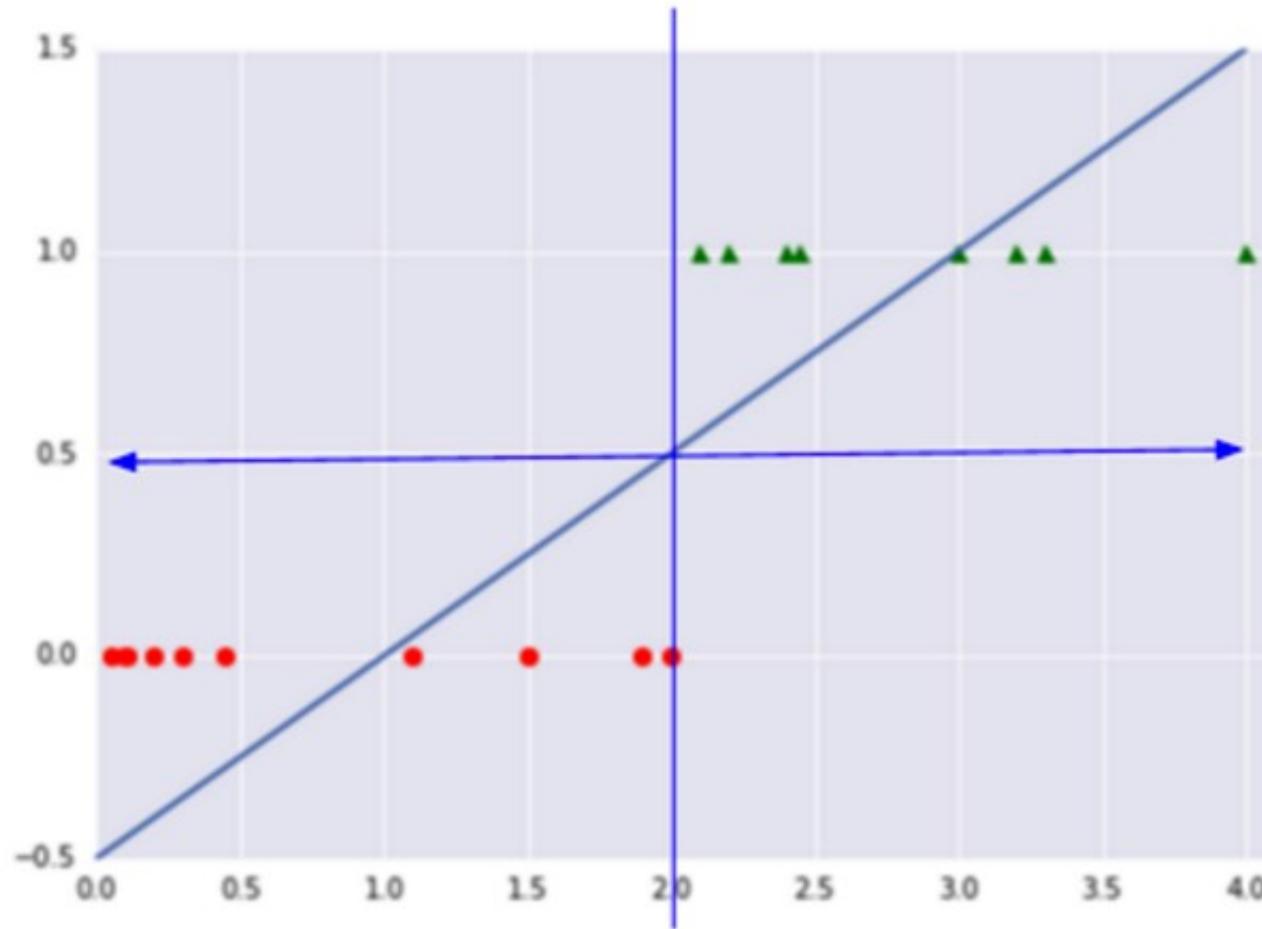
Binary Classification

- 종양의 크기에 따른 양성/음성 판별 문제
 - 1: 양성(암), 0: 음성(정상)



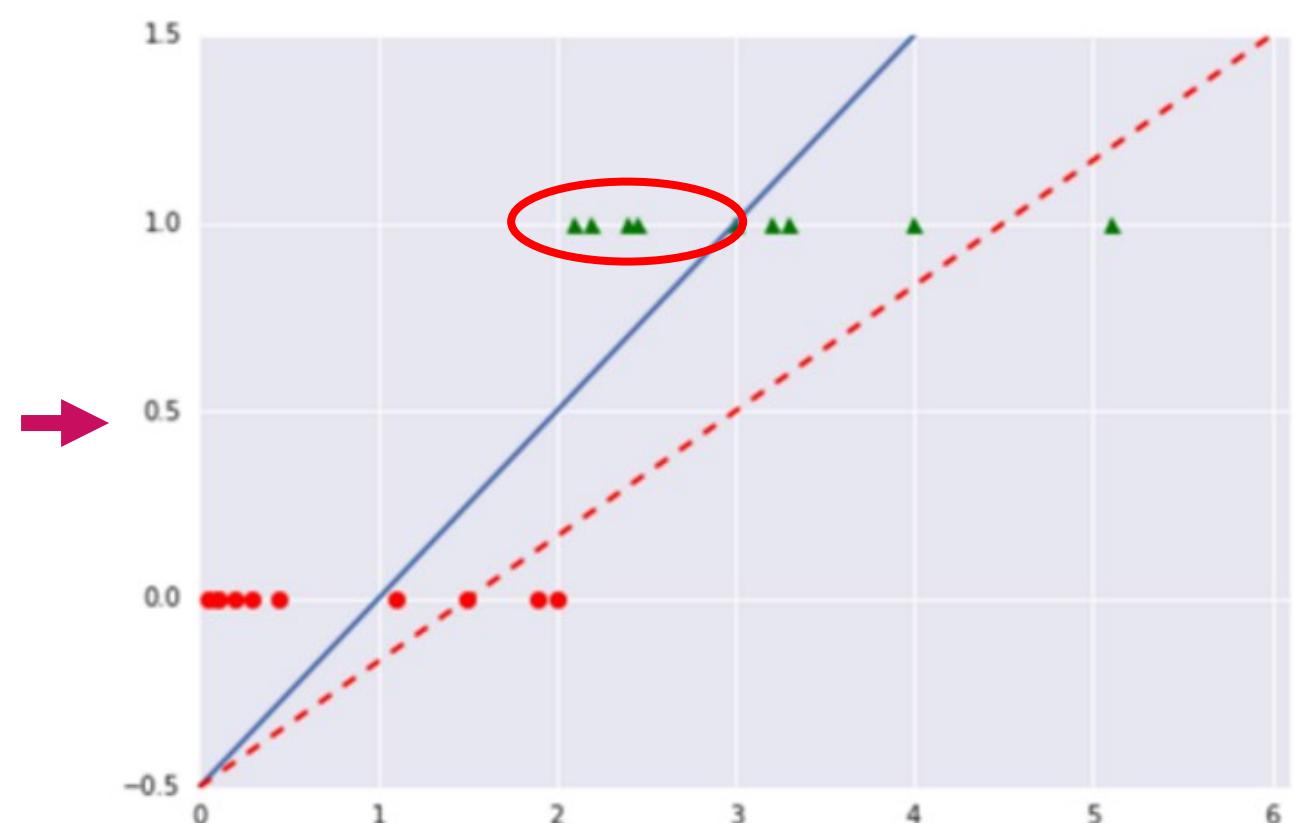
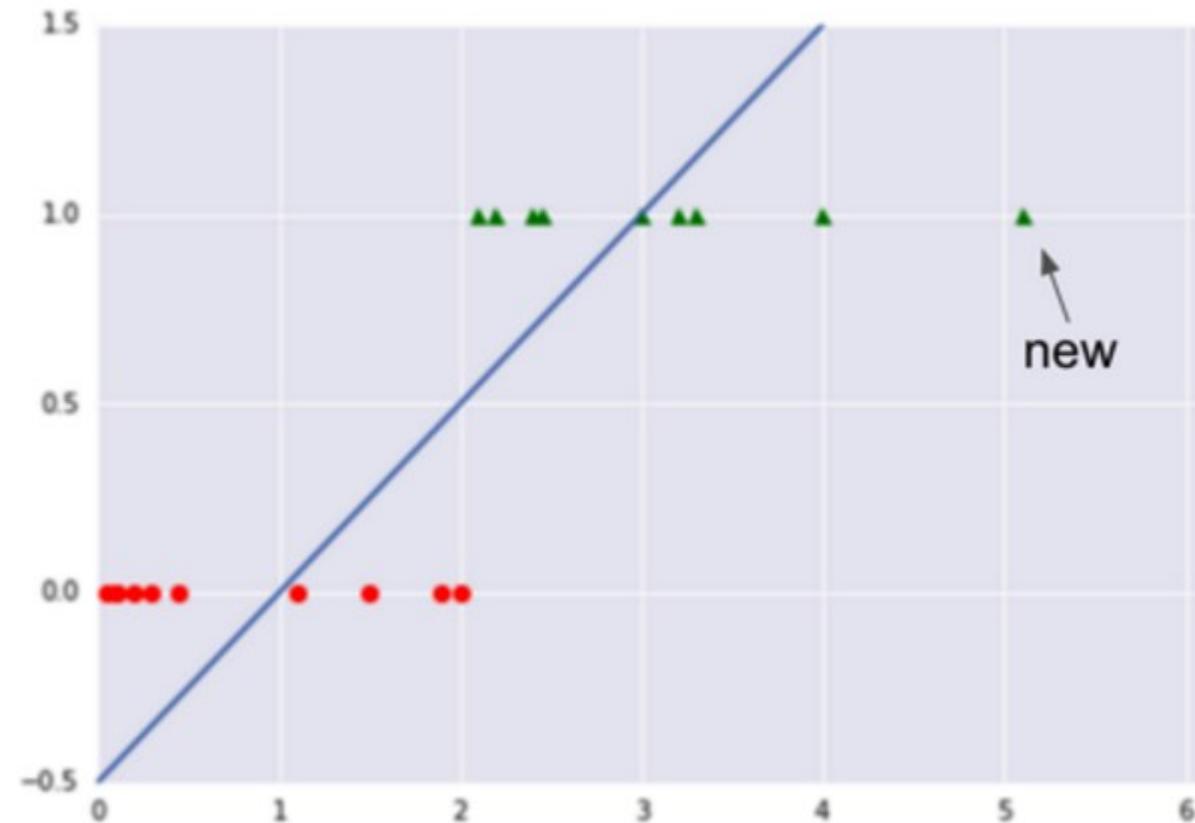
Binary Classification

- Linear Regression으로 해봅시다
 - Regression 예측값이 0.5 이상이면 양성, 0.5 이하면 음성으로 판별

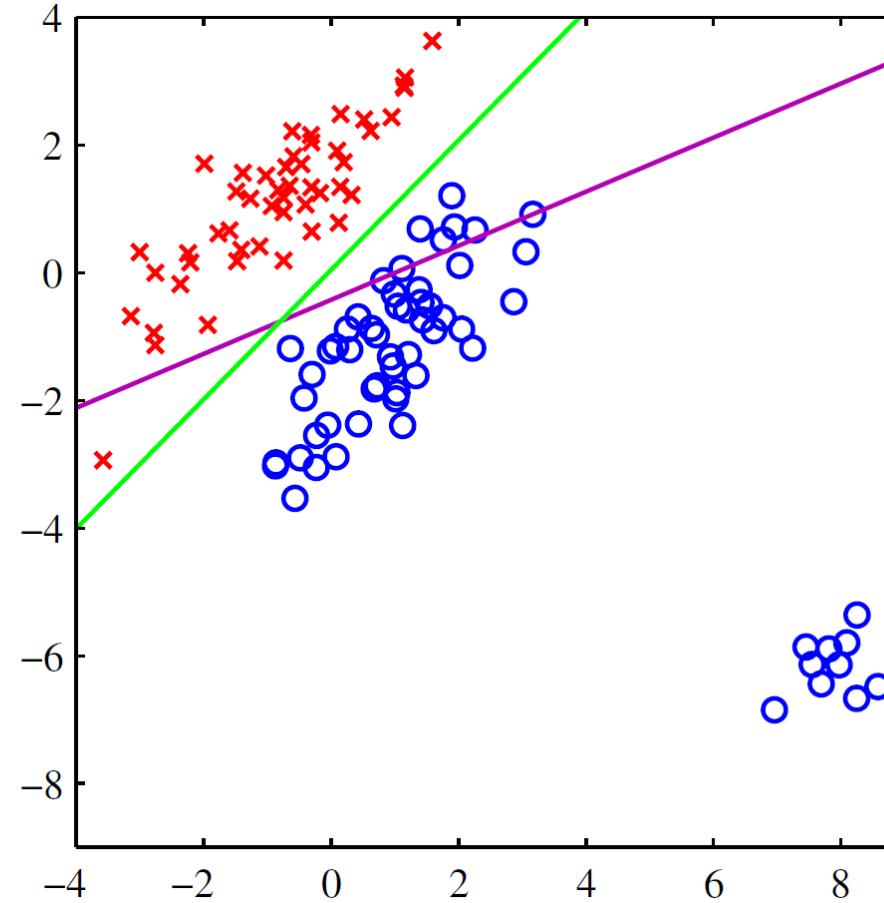
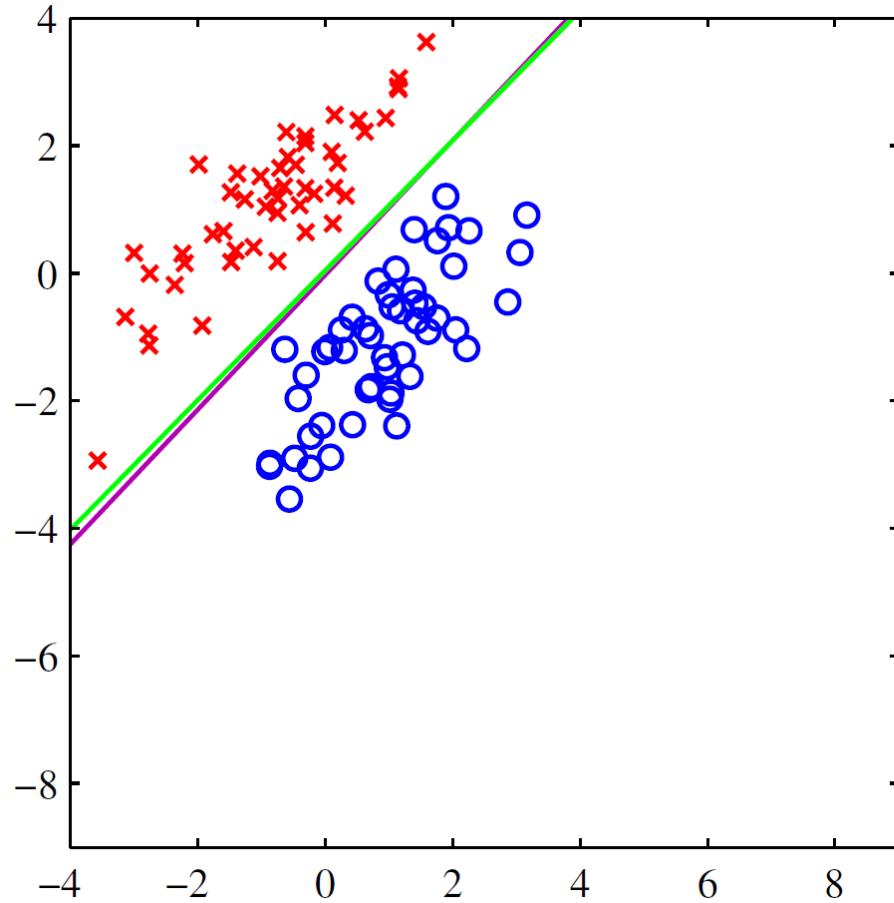


Binary Classification

- 종양의 크기가 매우 큰 data(outlier) 가 추가된 경우

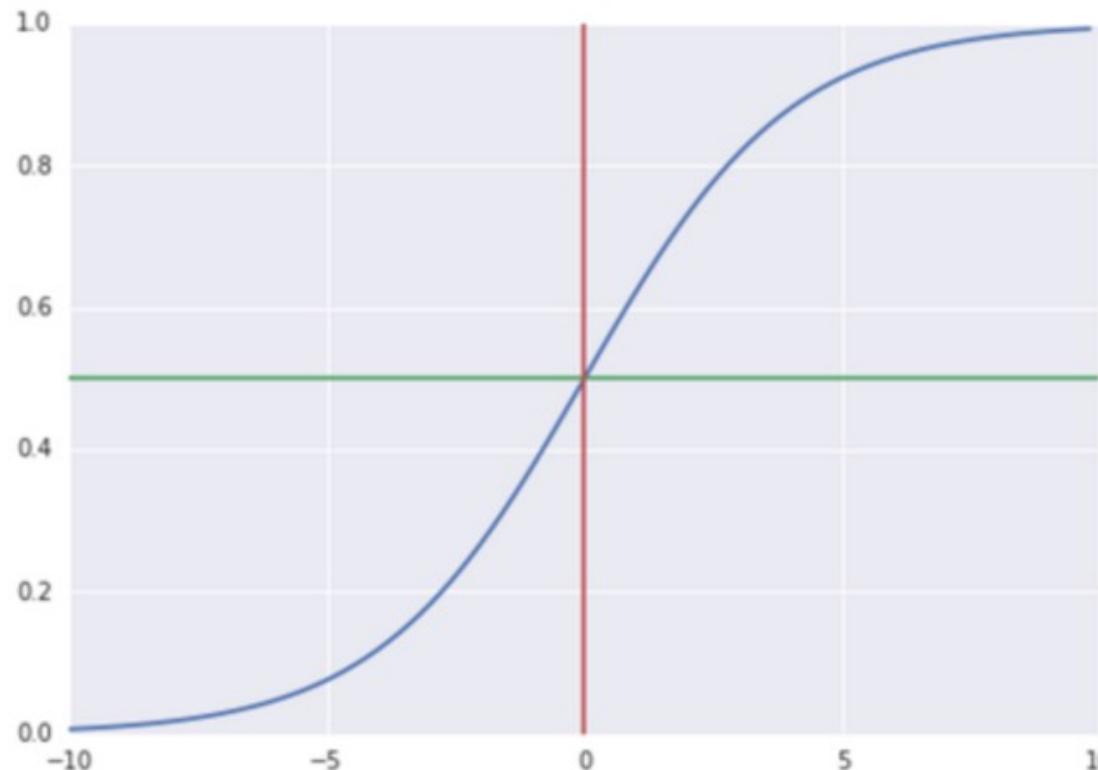


Problems of Linear Regression for Classification



Binary Classification

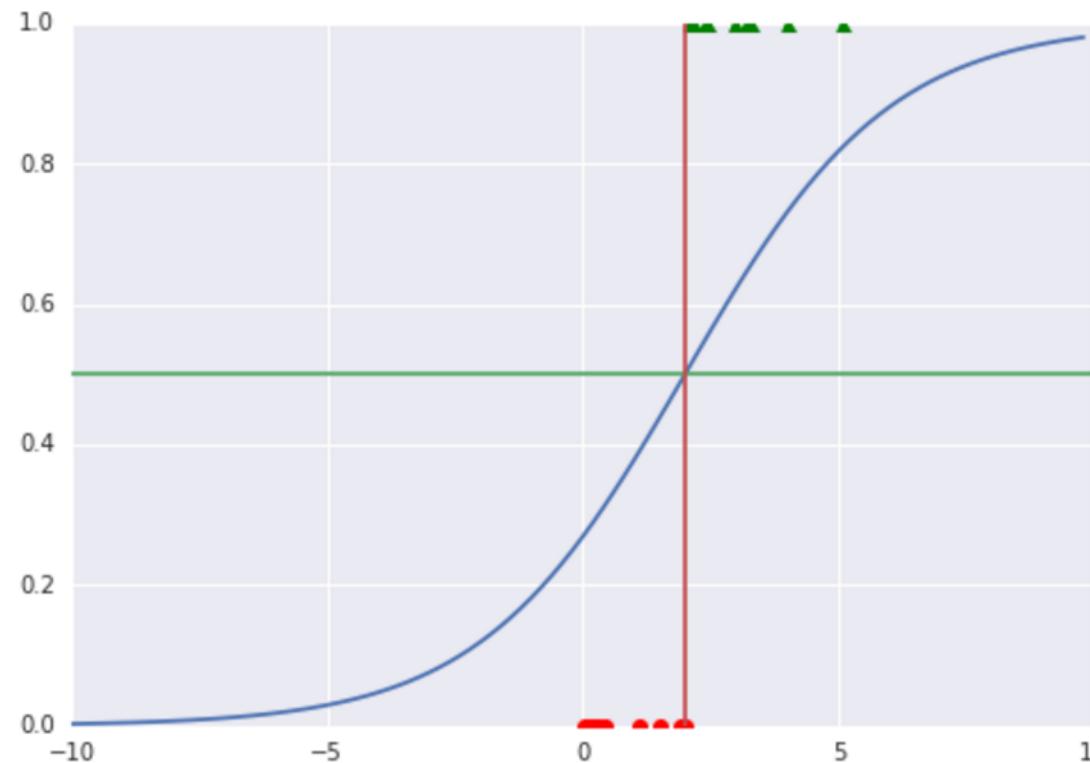
- 아주 크거나 아주 작은 data에 영향을 많이 받지 않았으면 좋겠다
- Binary classification에 맞게 0에서 1사이 값으로 나오면 좋겠다
→ Sigmoid 를 써보자



Logistic Regression

- Linear Regression 식에 Sigmoid 함수를 통과시킨 것

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



Logistic Regression

- 새로운 Cost(Loss) function을 정의(maximum likelihood estimation)

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

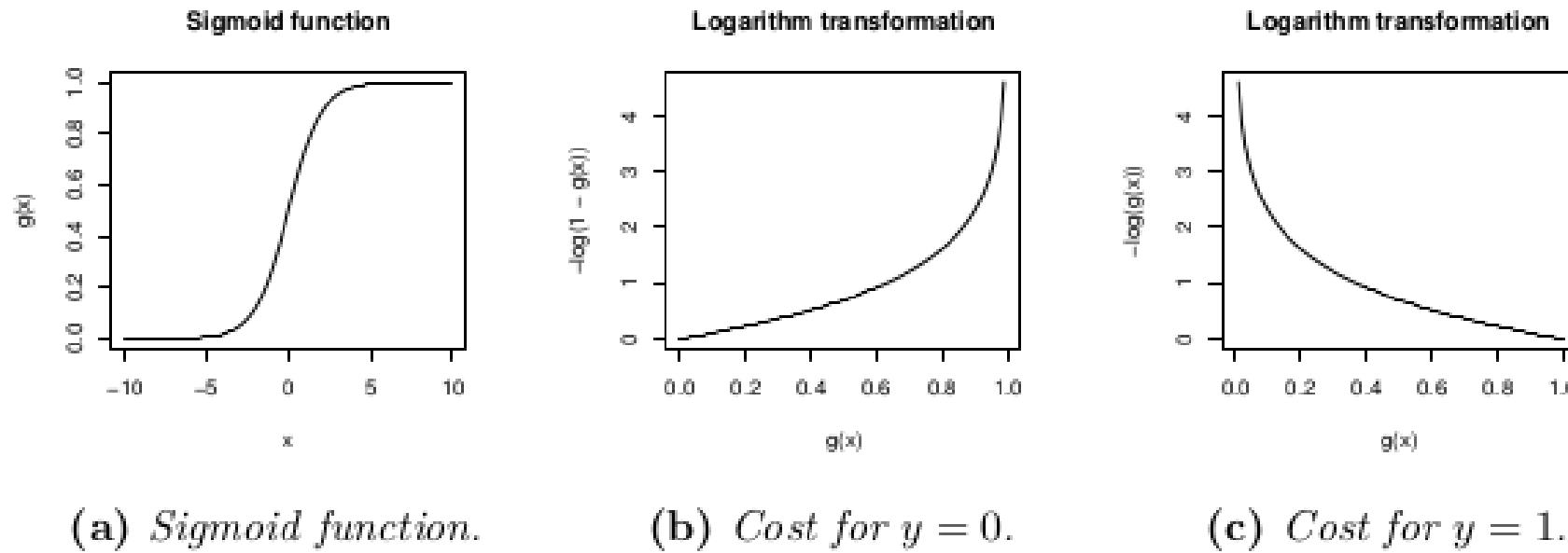


Figure B.1: Logarithmic transformation of the sigmoid function.

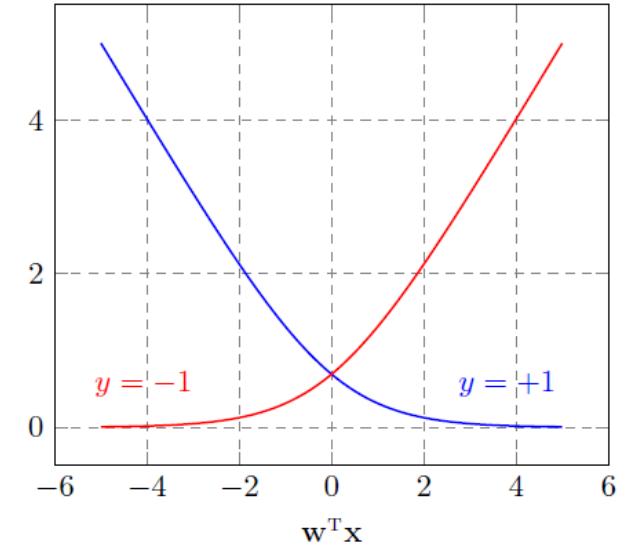
Minimizing NLL

$$\mathbf{e}(h(\mathbf{x}_n), y_n) = \ln \left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n} \right)$$

- We can define loss(error) function as below

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \log \left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n} \right)$$

$$\ln \left(1 + e^{-y \mathbf{w}^\top \mathbf{x}} \right)$$

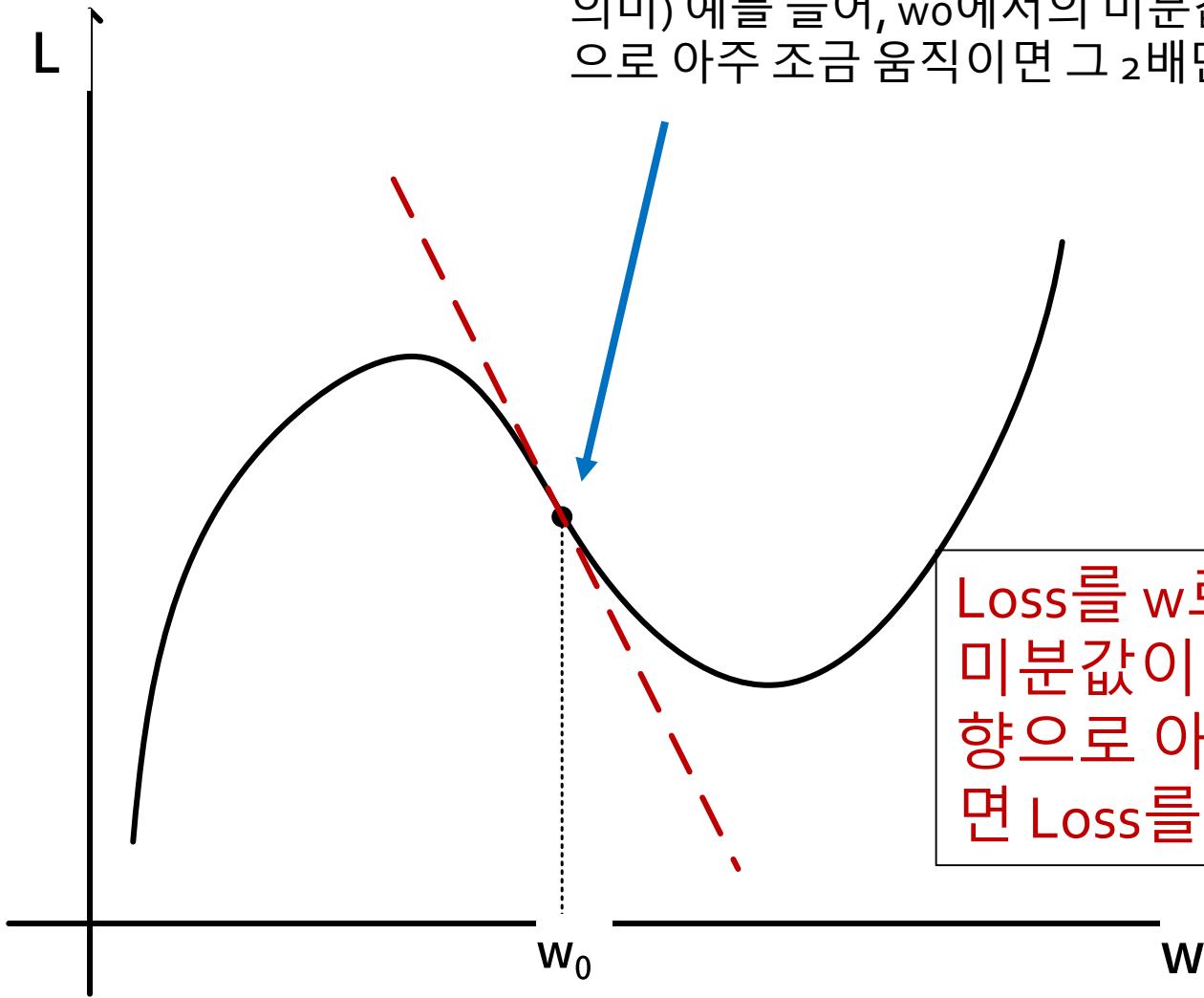


- Unfortunately, not easy to manipulate analytically

$$\begin{aligned}\nabla E_{\text{in}}(\mathbf{w}) &= -\frac{1}{N} \sum_{n=1}^N \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^\top \mathbf{x}_n}} \\ &= \frac{1}{N} \sum_{n=1}^N -y_n \mathbf{x}_n \theta(-y_n \mathbf{w}^\top \mathbf{x}_n)\end{aligned}$$

- We need **iterative** optimization
- Use 미분!

미분??



$w=w_0$ 에서의 미분값

= 이 점에서의 접선의 기울기

의미) 예를 들어, w_0 에서의 미분값이 -2 라면, w 를 w_0 에서 왼쪽(-방향)
으로 아주 조금 움직이면 그 2배만큼 L 값이 증가한다는 뜻!

Loss를 w 로 미분하고,
미분값이 가리키는 방향의 반대방
향으로 아주 조금씩 w 를 바꿔나가
면 Loss를 감소시킬 수 있다!!

Gradient Descent

Loss Function의 미분(Gradient)를 이용하여 weight를 update하는 방법

$$w_{new} = w_{old} - \eta \nabla_w L$$

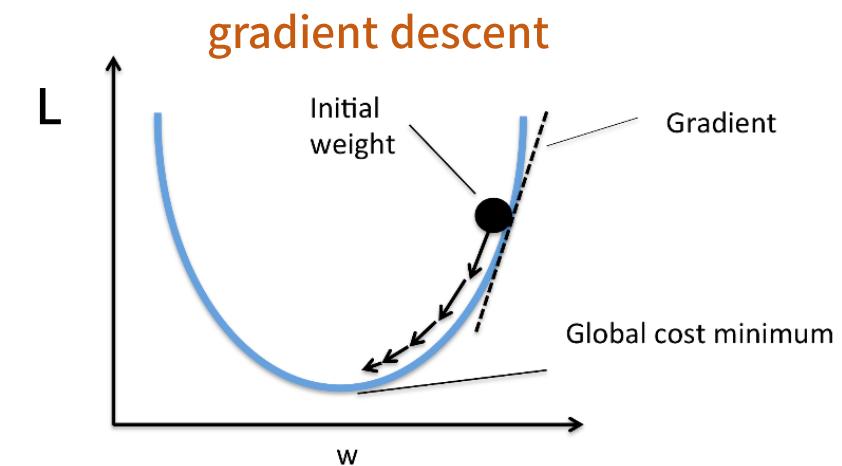
Weight update = $w_{new} - w_{old}$

$$= -\eta \nabla_w L$$

Loss를 감소 시키는 방향
(Descent)

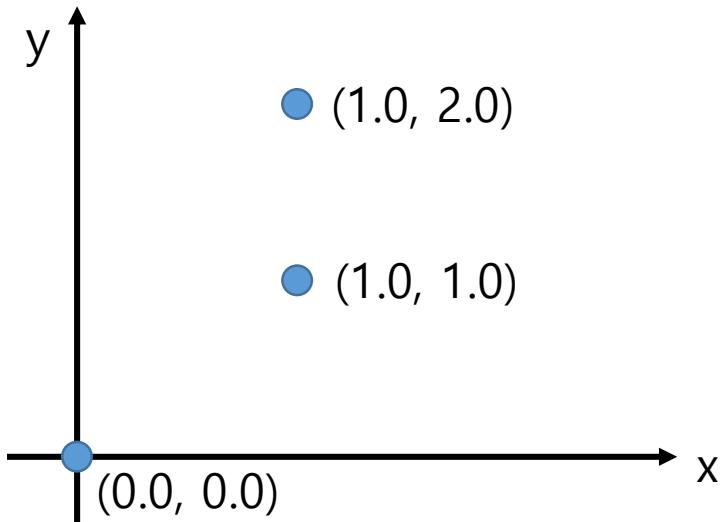
아주 조금씩 이동
(Learning Rate)

미분값
(Gradient)



Linear Regression Again

- Find the linear function(f) that best describes the given data
 - $H(x, w_0, w_1) = w_1 x + w_0$



$$\begin{aligned} L &= \sum_i (y_i - w_1 x_i - w_0)^2 \\ &= (0.0 - w_1 \cdot 0.0 - w_0)^2 + (1.0 - w_1 \cdot 1.0 - w_0)^2 \\ &\quad + (2.0 - w_1 \cdot 1.0 - w_0)^2 \\ &= 2w_1^2 + 3w_0^2 - 6w_1 - 6w_0 + 4w_1 w_0 + 5 \end{aligned}$$

Solving Linear Regression Using GD

- Choose a small value for η such as $\eta = 0.1$
- Randomly select $w_0^0 = 1, w_1^0 = 1$, initially
- Repeat

$$w_0^{t+1} = w_0^t - \eta(4w_1^t + 6w_0^t - 6)$$

$$w_1^{t+1} = w_1^t - \eta(4w_1^t + 4w_0^t - 6)$$

$$\frac{\partial L}{\partial w_1} = 4w_1 + 4w_0 - 6 = 0$$

$$\frac{\partial L}{\partial w_0} = 4w_1 + 6w_0 - 6 = 0$$

Solving Linear Regression Using GD

$$w_0^0 = 1$$

$$w_1^0 = 1$$

$$w_0^1 = 1 - 0.1(4 \times 1 + 6 \times 1 - 6) = 0.6$$

$$w_1^1 = 1 - 0.1(4 \times 1 + 4 \times 1 - 6) = 0.8$$

$$w_0^2 = 0.6 - 0.1(4 \times 0.8 + 6 \times 0.6 - 6) = 0.54$$

$$w_1^2 = 0.8 - 0.1(4 \times 0.8 + 4 \times 0.6 - 6) = 0.84$$

$$w_0^3 = 0.54 - 0.1(4 \times 0.84 + 6 \times 0.54 - 6) = 0.480$$

$$w_1^3 = 0.84 - 0.1(4 \times 0.84 + 4 \times 0.54 - 6) = 0.888$$

Solving Linear Regression Using GD

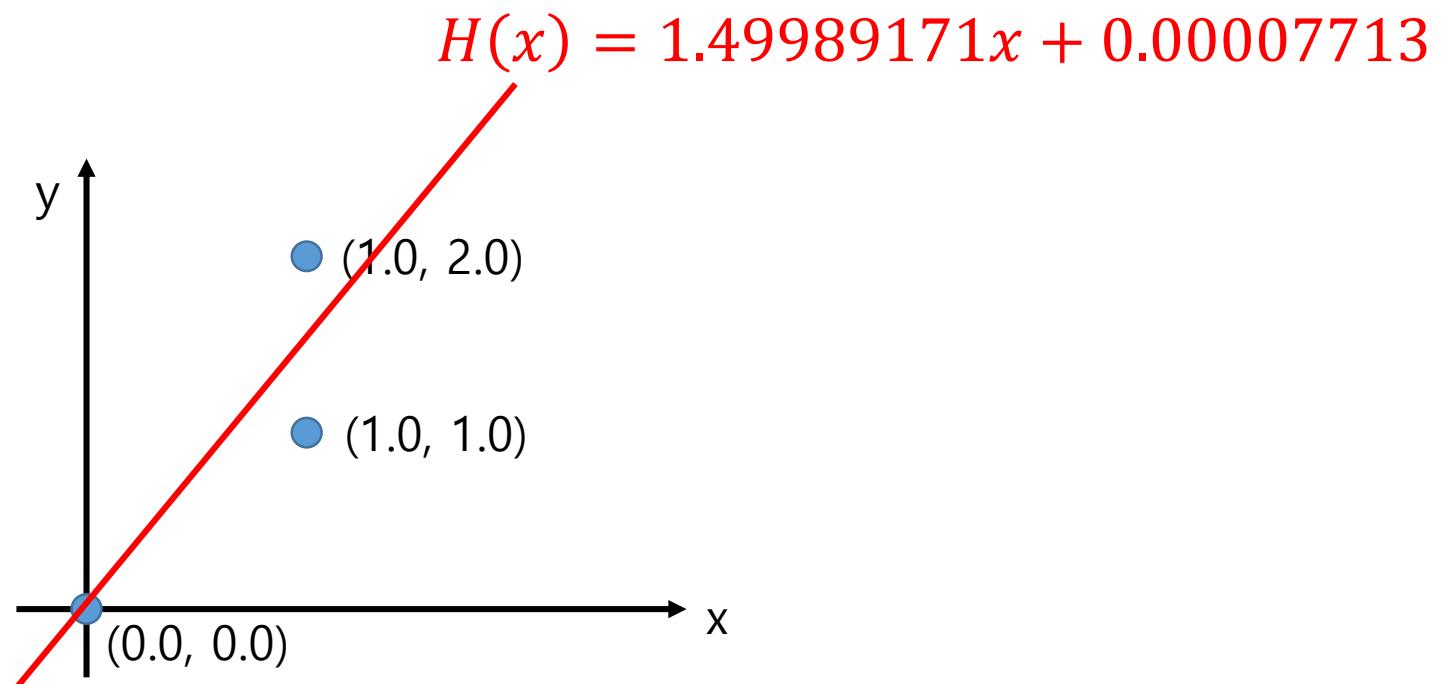
$$w_0^4 = 0.480 - 0.1(4 \times 0.888 + 6 \times 0.480 - 6) = 0.4368$$

$$w_1^4 = 0.888 - 0.1(4 \times 0.888 + 4 \times 0.480 - 6) = 0.9408$$

...

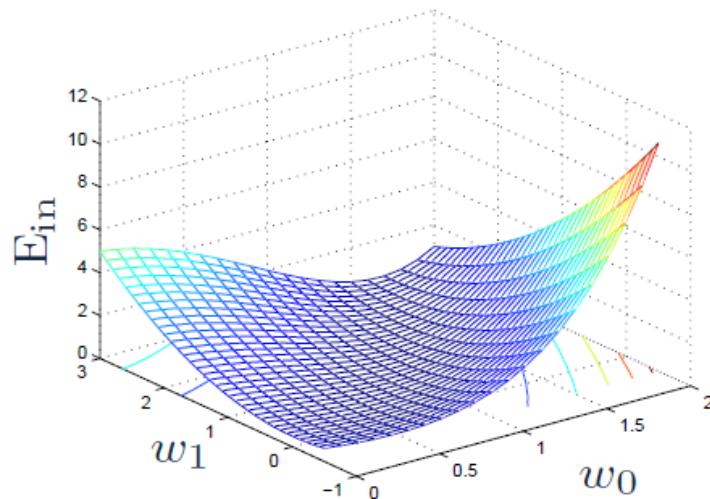
$$w_0^{100} = 0.00007713$$

$$w_1^{100} = 1.49989171$$



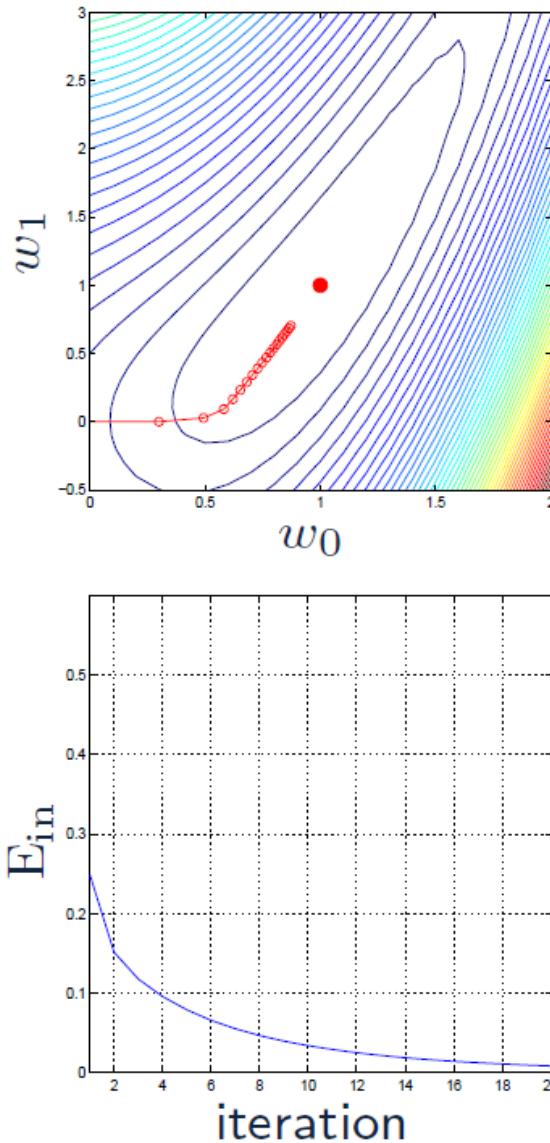
Gradient Decent Example

► Global minimum 0 at (1,1)



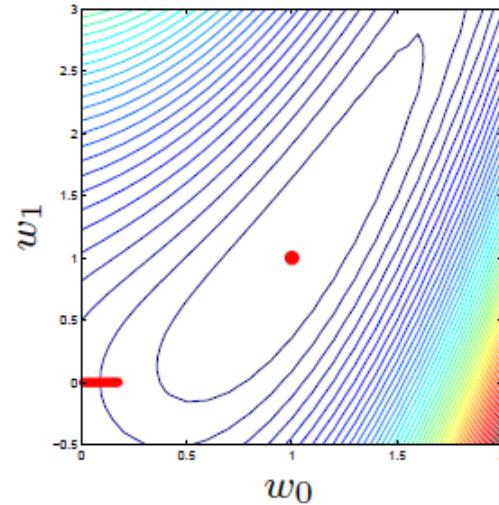
► Start at (0,0)

- # iterations (steps) = 20
- step size $\eta = 0.3$

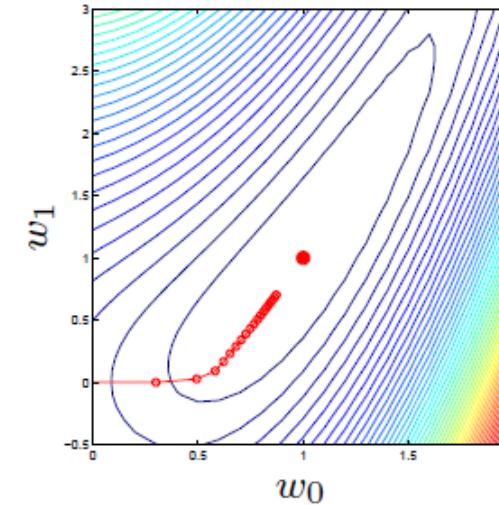


Learning Rate

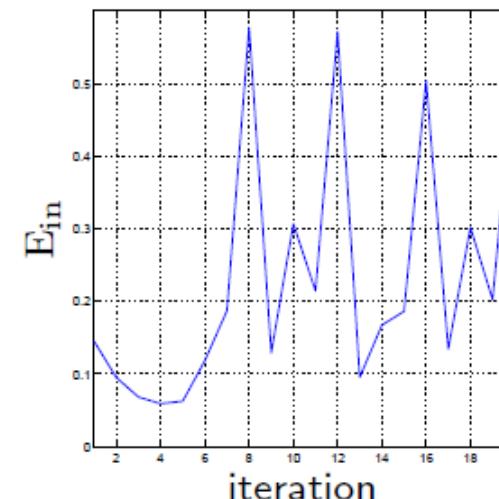
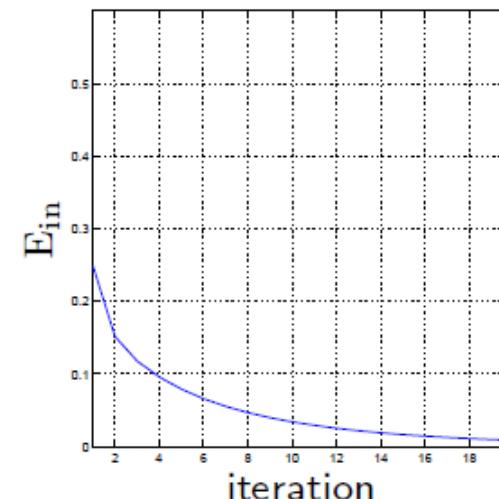
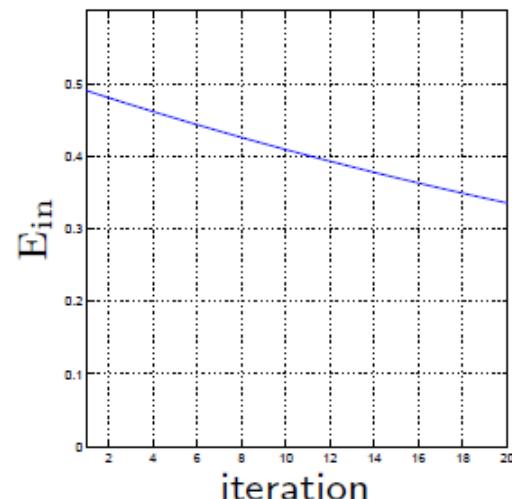
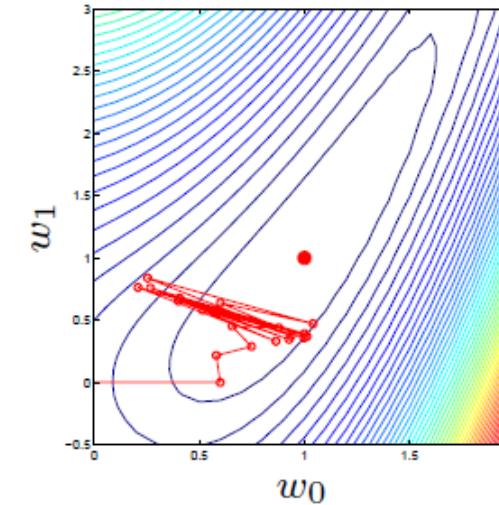
$$\eta = 0.01$$



$$\eta = 0.3$$

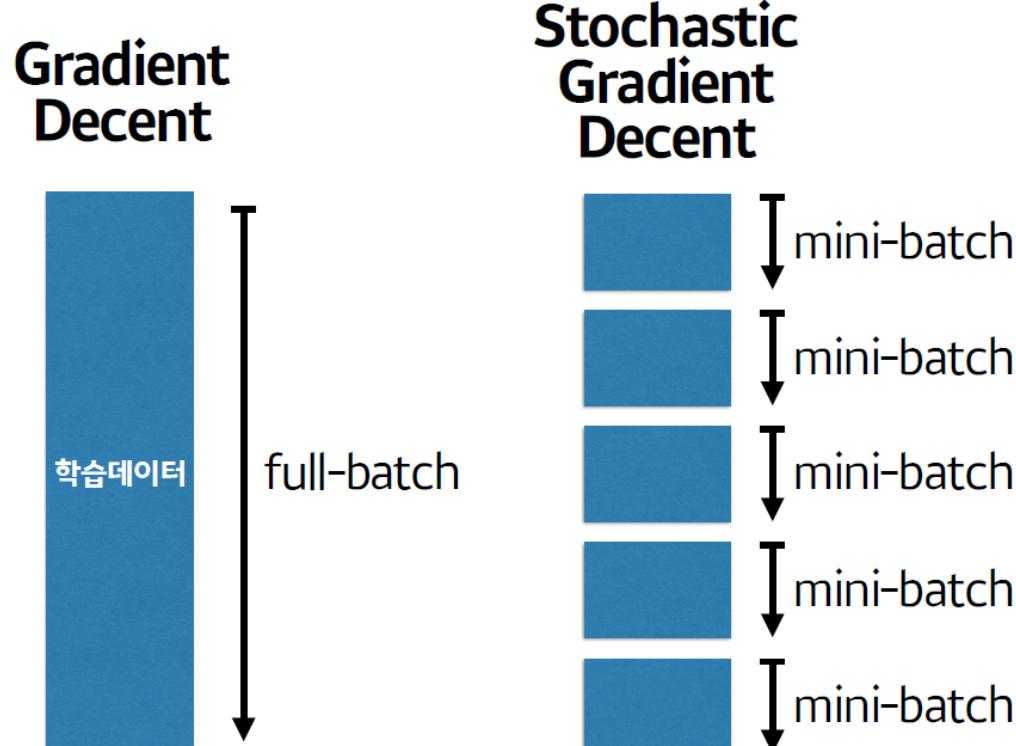


$$\eta = 0.6$$



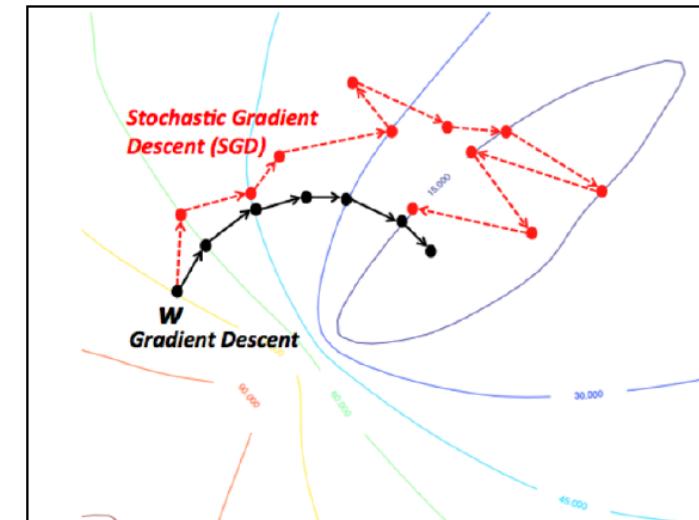
Stochastic Gradient Descent, Mini-batch Training

- Data가 너무 많아서 한번에 다 넣고 학습하면 시간도 오래걸리고, memory도 부족하게 됨



전부다 읽고나서
최적의 1스텝 간다.

작은 토막마다
일단 1스텝간다.



Learning Rate & Mini-Batch



Mini-Batch size: Number of training instances the network evaluates per weight update step.

- Larger batch size = more computational speed
- Smaller batch size = (empirically) better generalization

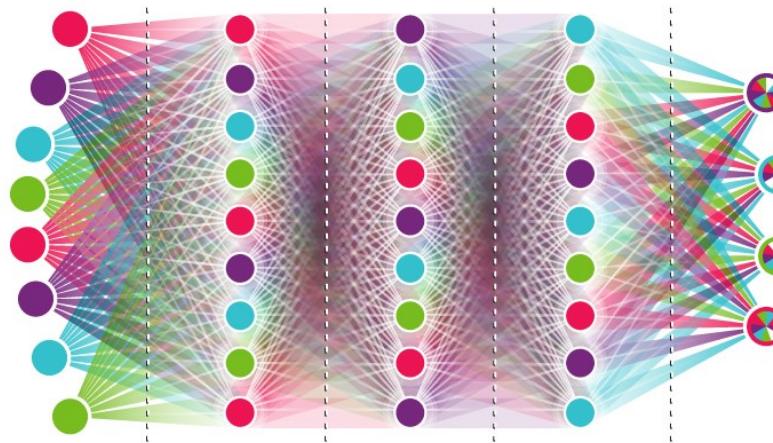
“Training with large minibatches is bad for your health. More importantly, it's bad for your test error. Friends don't let friends use minibatches larger than 32.”

- Yann LeCun

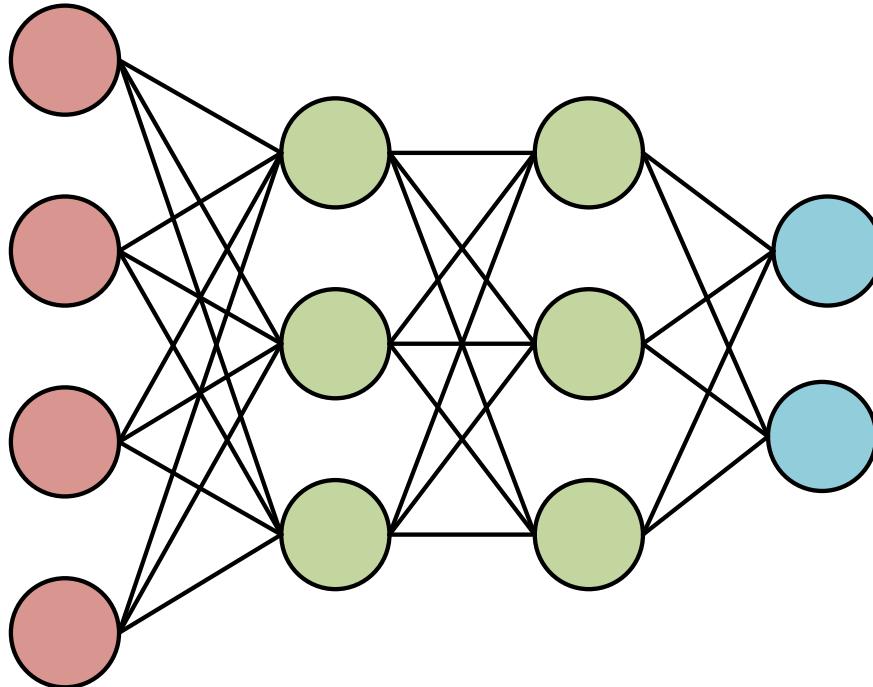
[Revisiting Small Batch Training for Deep Neural Networks](#) (2018)

“when increasing the batch size, a linear increase of the learning rate η with the batch size m is required to keep the mean SGD weight update per training example constant”

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 개일 때는
어떻게 학습할 수 있을까?

먼저 Multi-Layer부터 생각해봅시다

미분을 계산해봅시다!

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

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

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

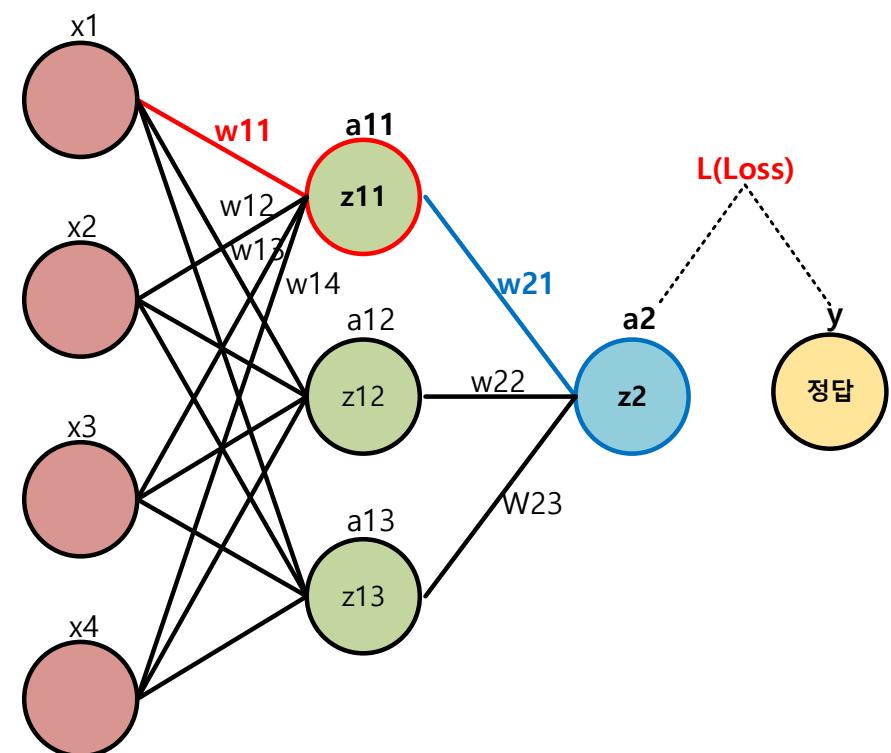
$$a_2 = z_2$$

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

일 때,

w₁₁을 update 하기 위해 필요한 미분값

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



Back Propagation

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

Loss부터 거꾸로 한 단계씩 미분을 해봅시다

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

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

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

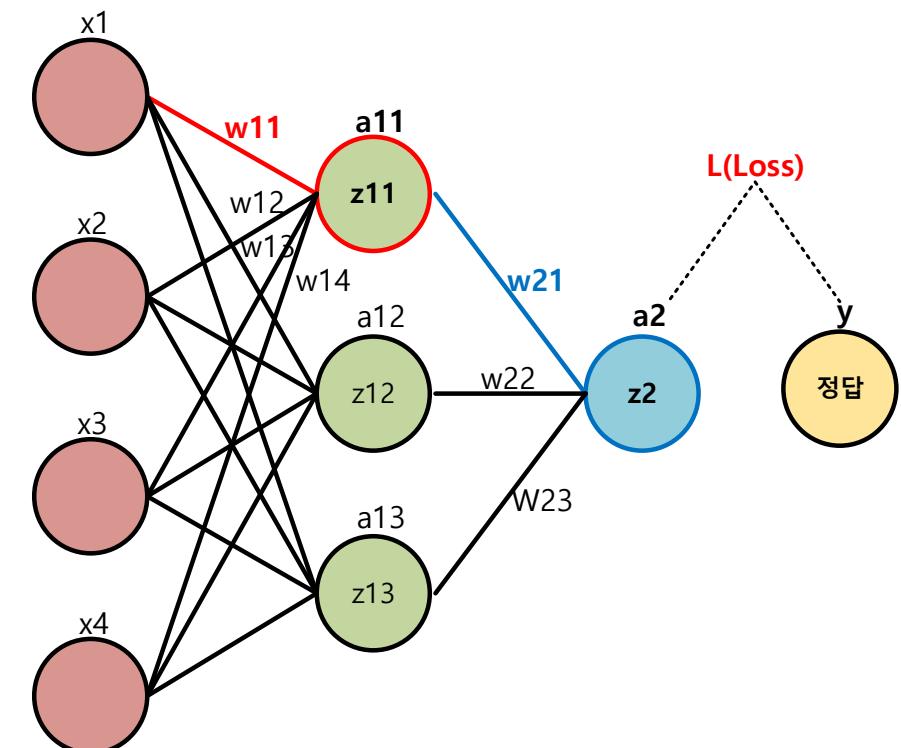
$$\partial a_{11} / \partial z_{11} = \sigma(z_{11}) \cdot (1 - \sigma(z_{11}))$$

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

이 미분들을 전부 각각 곱하면(chain rule),

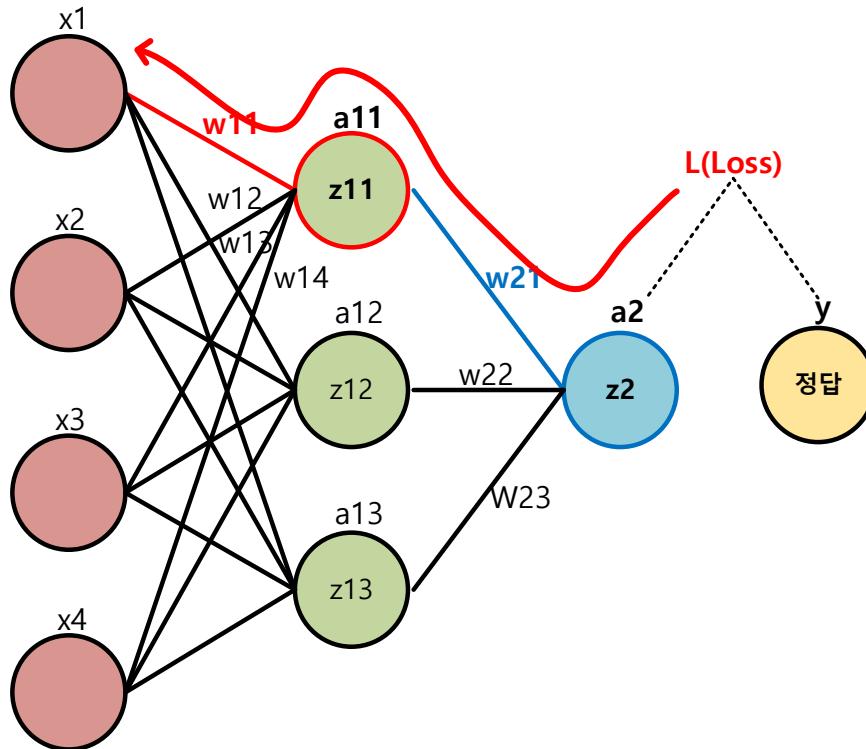
$$\frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{11}} = \frac{\partial L}{\partial w_{11}}$$

우리가 구하려고 했던 미분값



Back Propagation

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

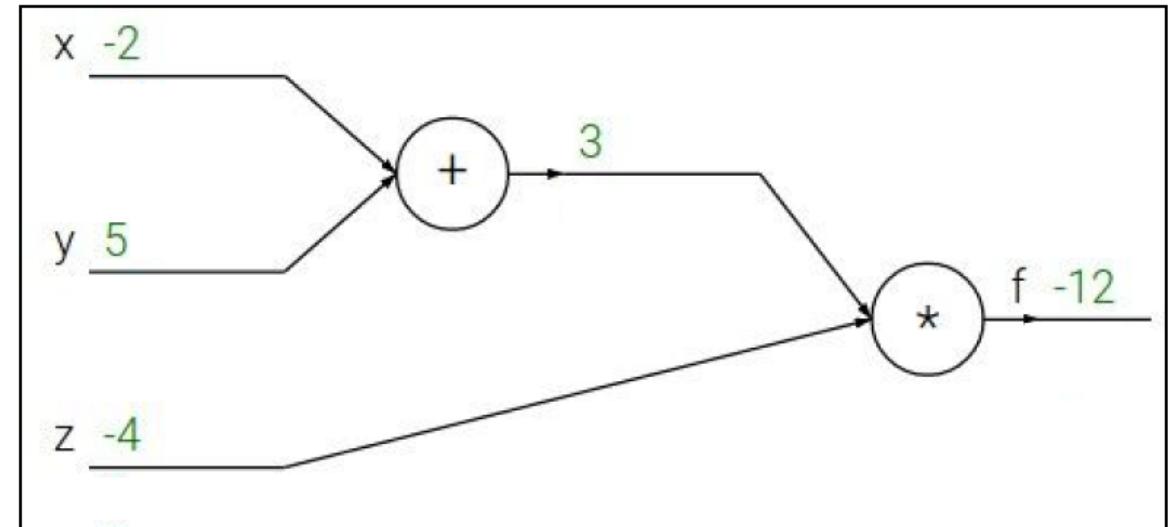


Back Propagation

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

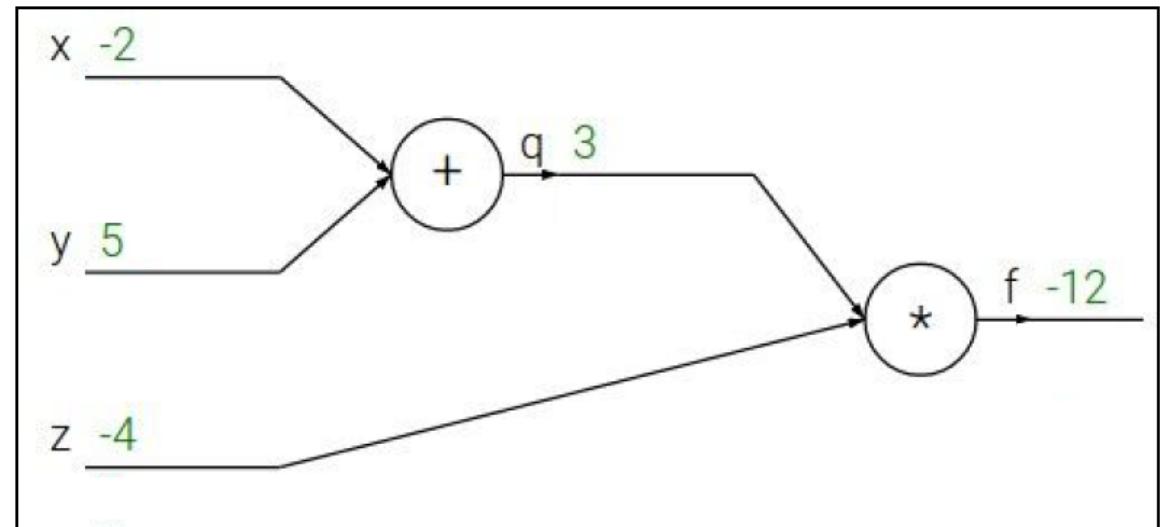
$$f(x, y, z) = (x + y)z$$

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

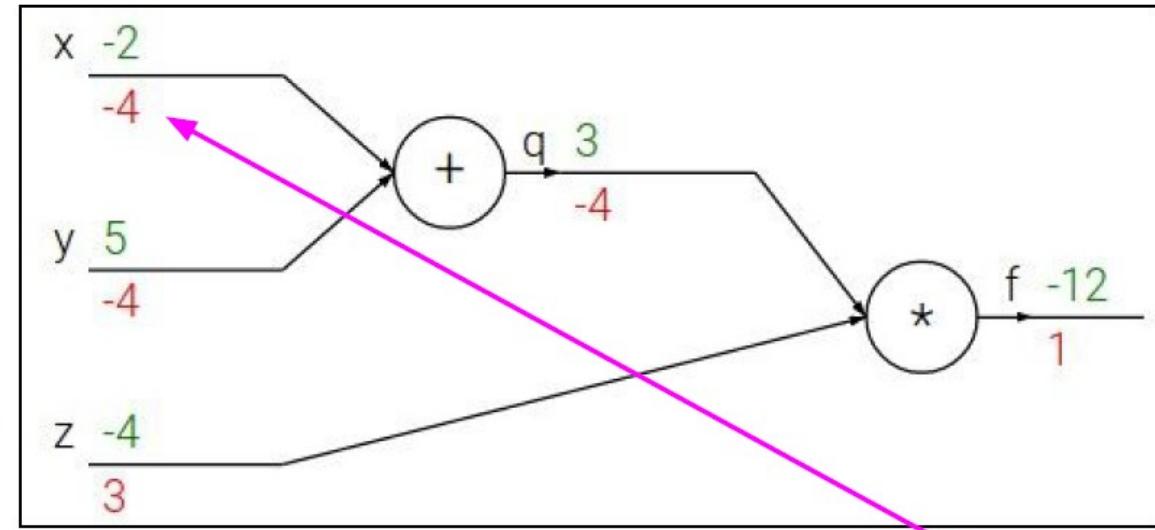
$$f(x, y, z) = (x + y)z$$

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

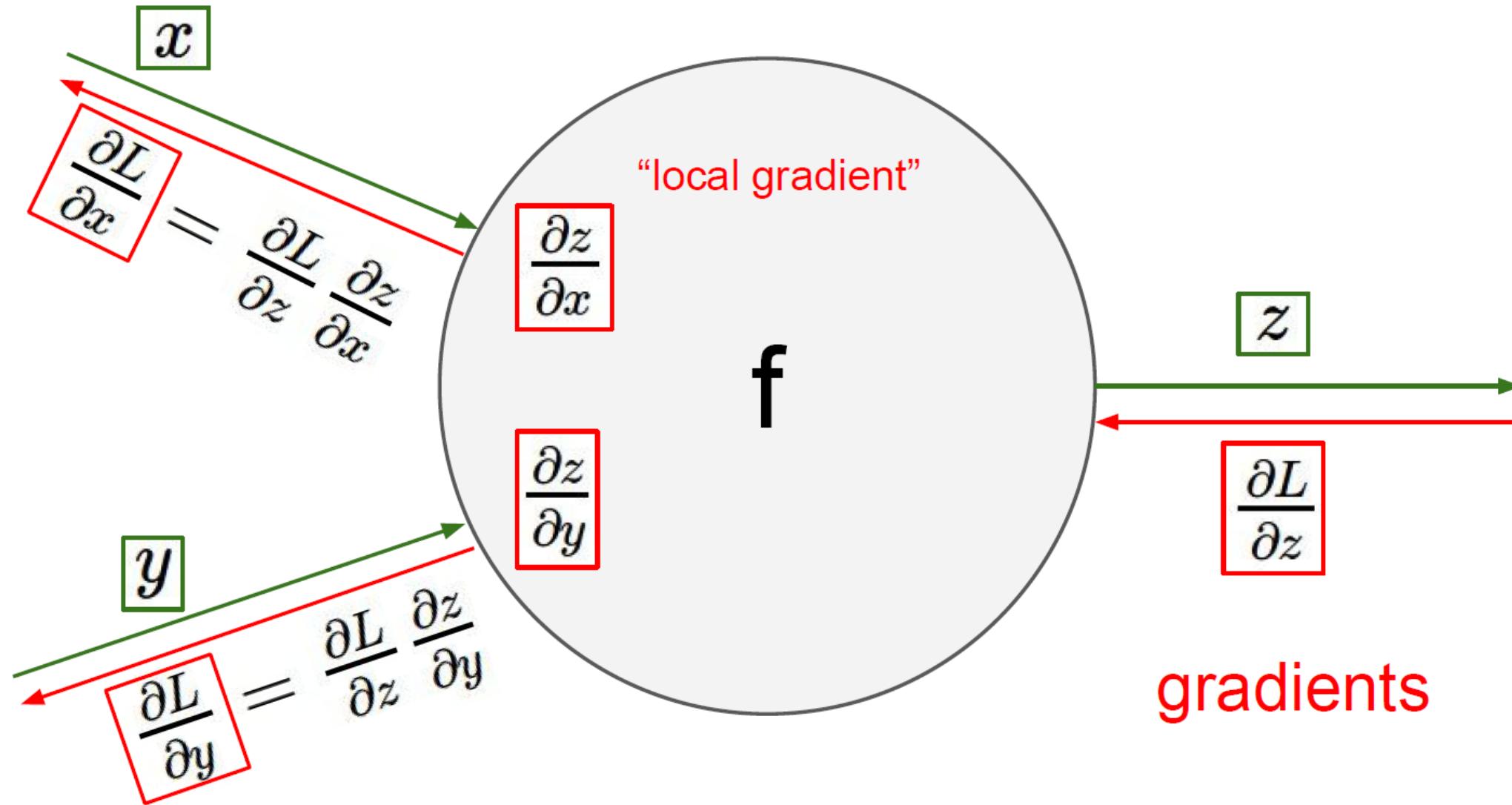


Chain rule:

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

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

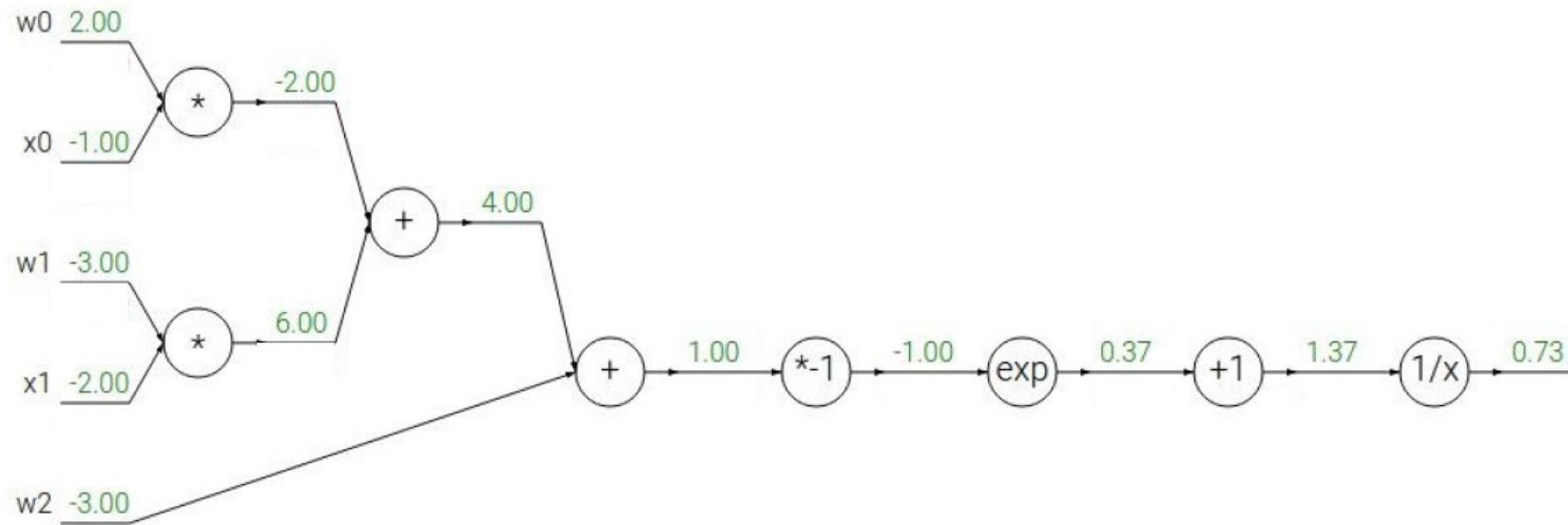
Chain Rule(Local Gradient)



Back Propagation(Example)

Another example:

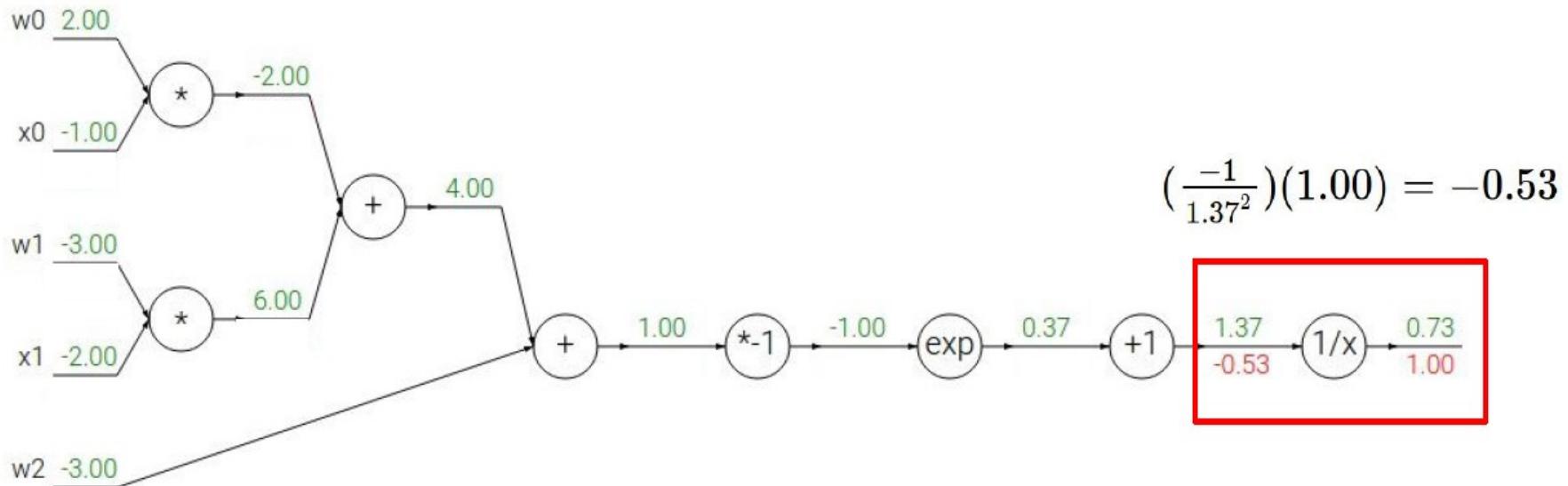
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Back Propagation(Example)

Another example:

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



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

→

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

$$f_a(x) = ax$$

→

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

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

→

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

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

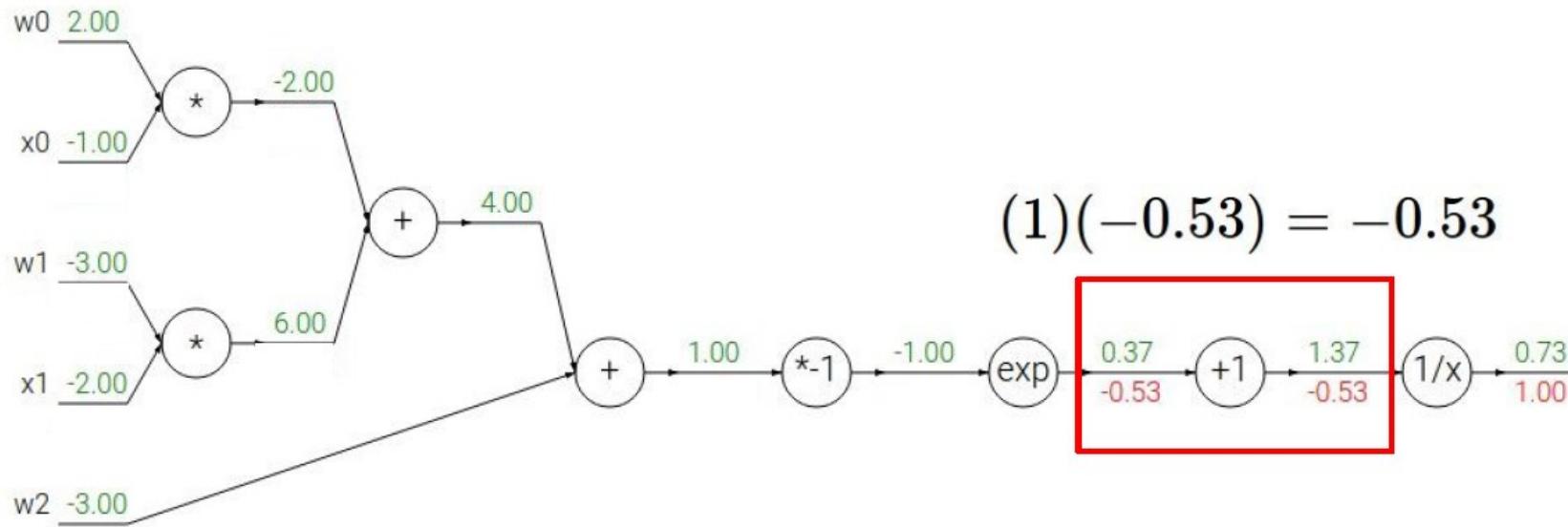
→

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

Back Propagation(Example)

Another example:

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



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

→

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

$$f_a(x) = ax$$

→

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

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

→

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

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

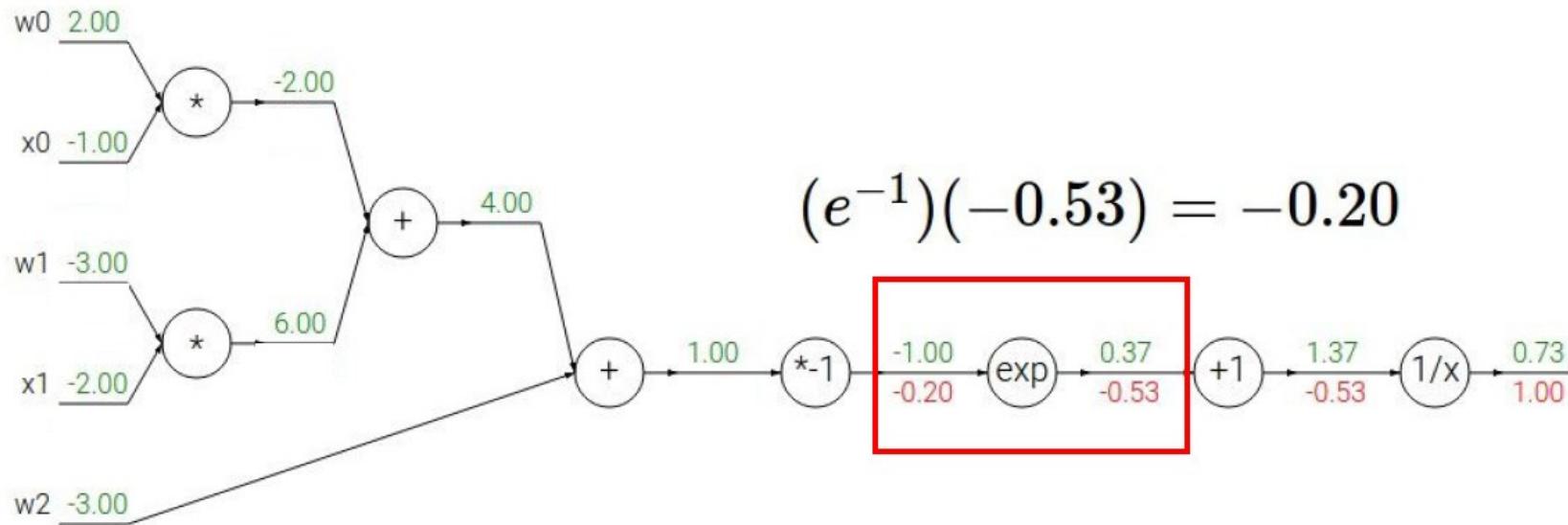
→

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

Back Propagation(Example)

Another example:

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



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

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

→

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

$$f_a(x) = ax$$

→

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

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

→

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

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

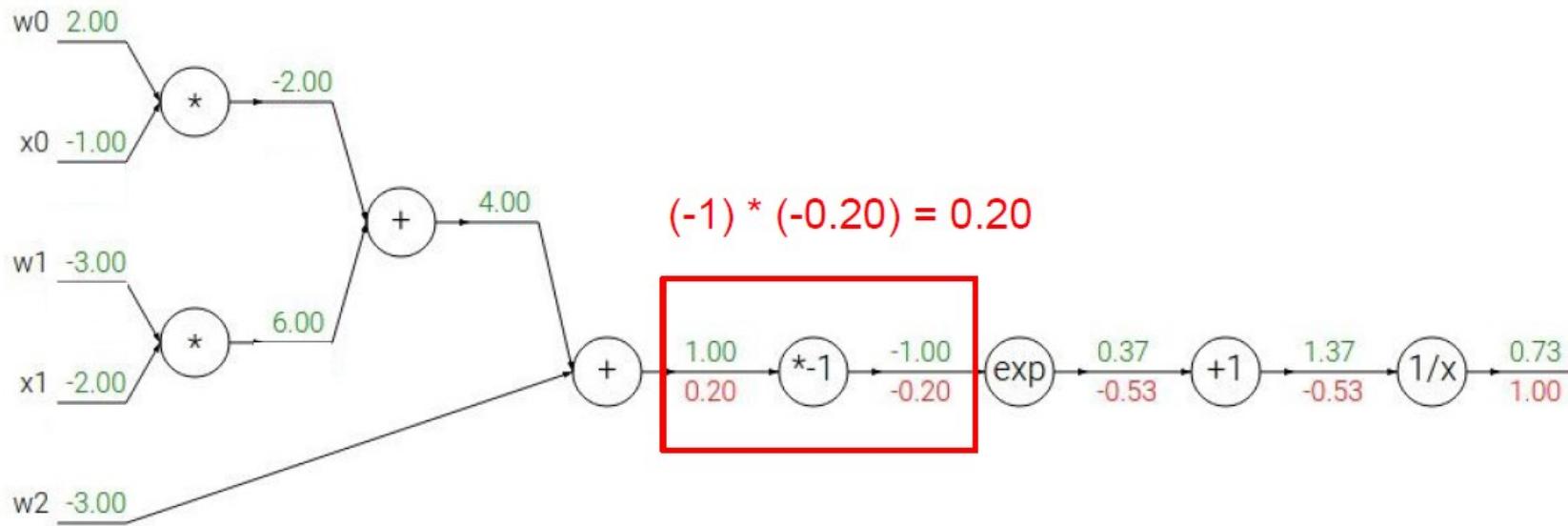
→

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

Back Propagation(Example)

Another example:

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



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

\rightarrow

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

$$f_a(x) = ax$$

\rightarrow

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

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

\rightarrow

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

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

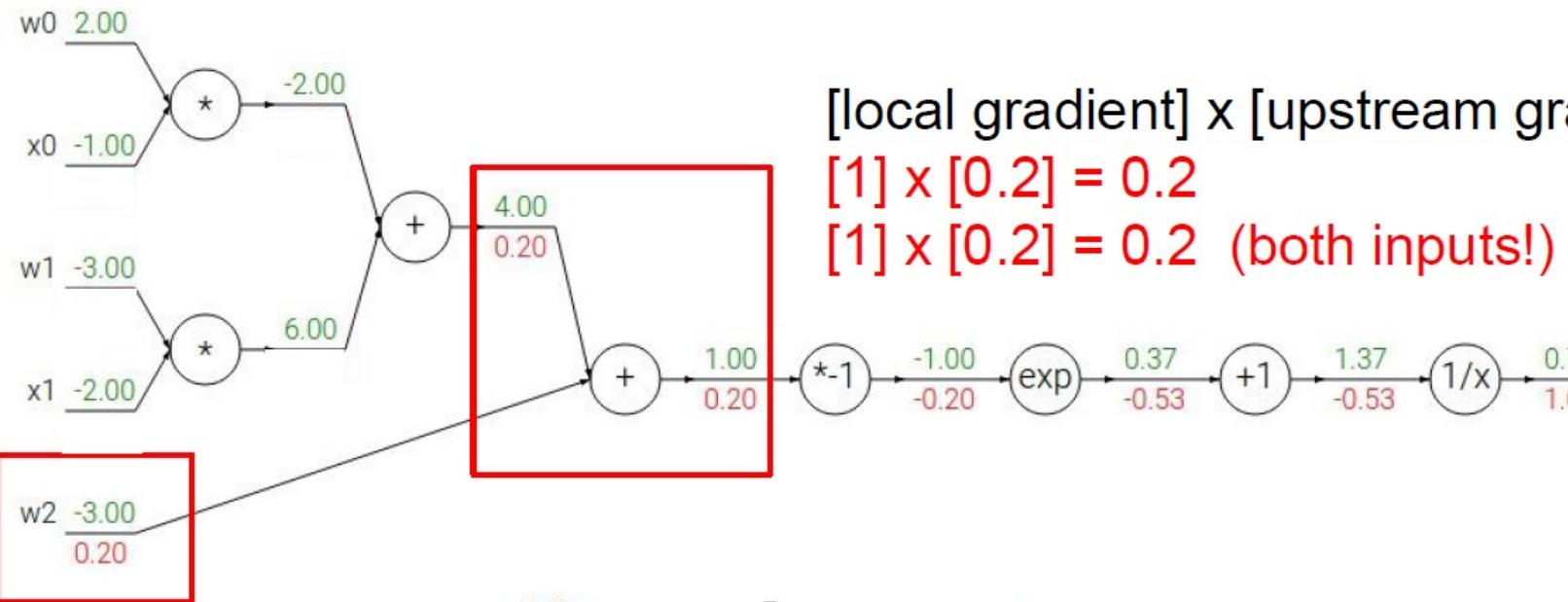
\rightarrow

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

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

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

$$f_a(x) = ax$$

\rightarrow

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

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

\rightarrow

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

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

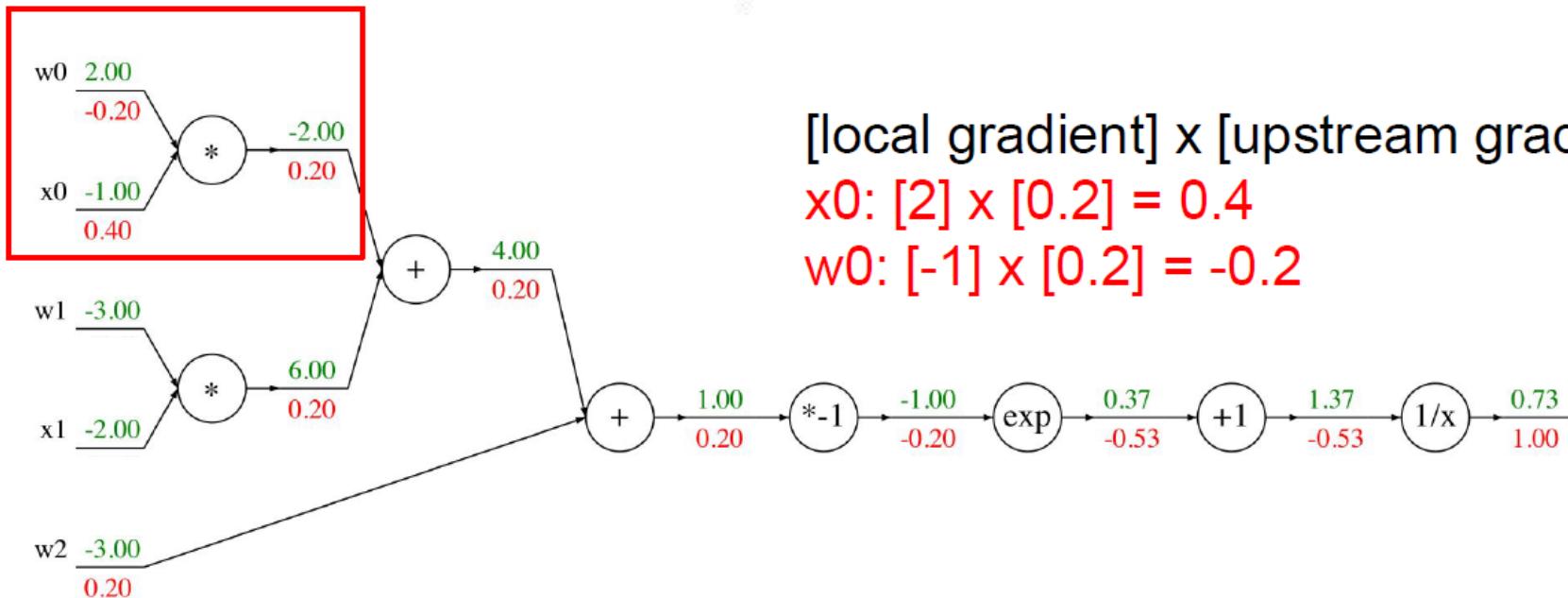
\rightarrow

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

Back Propagation(Example)

Another example:

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



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

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

→

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

$$f_a(x) = ax$$

→

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

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

→

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

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

→

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

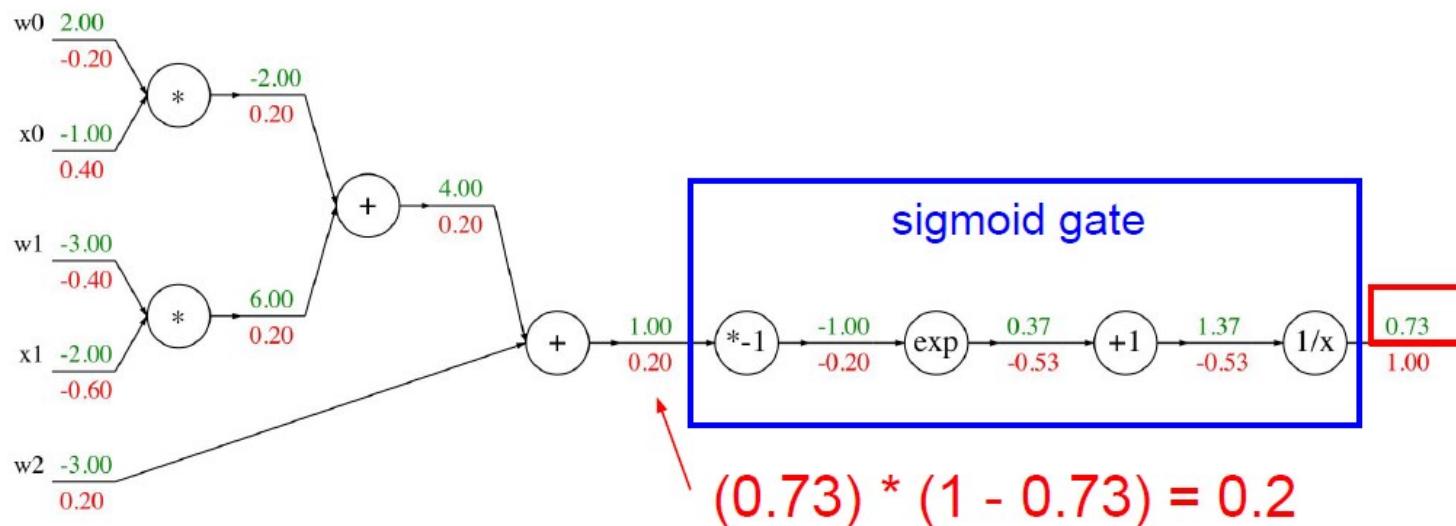
Back Propagation(Example)

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

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

sigmoid function

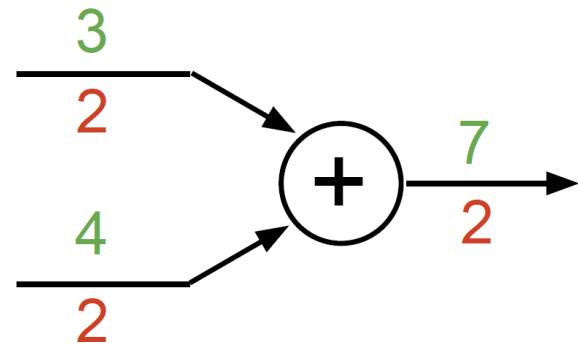
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left(\frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$



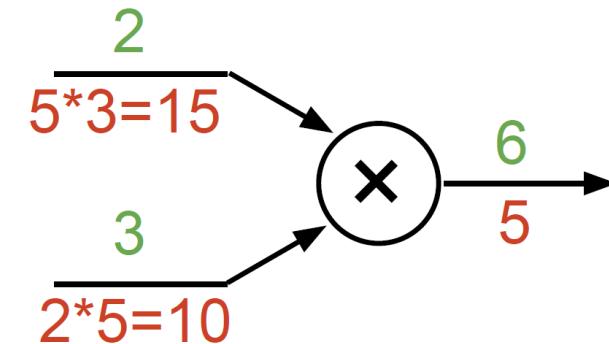
Back Propagation(Example)

Patterns in gradient flow

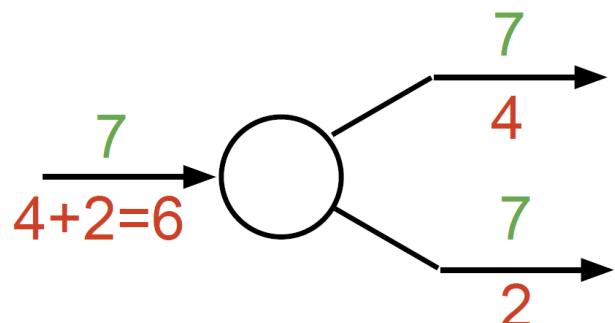
add gate: gradient distributor



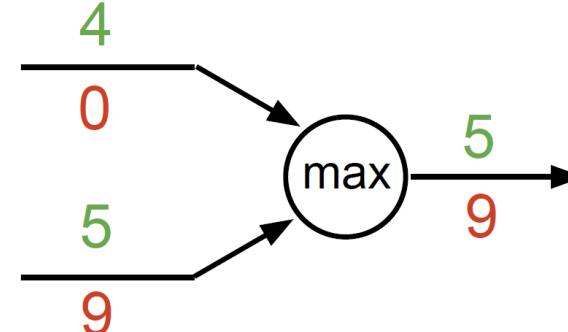
mul gate: “swap multiplier”



copy gate: gradient adder



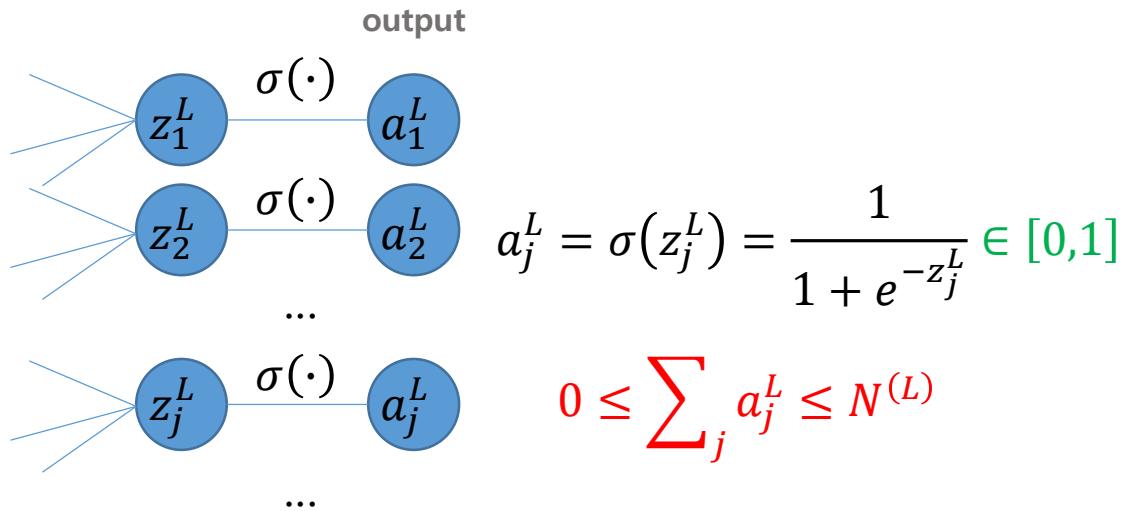
max gate: gradient router



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

Softmax!

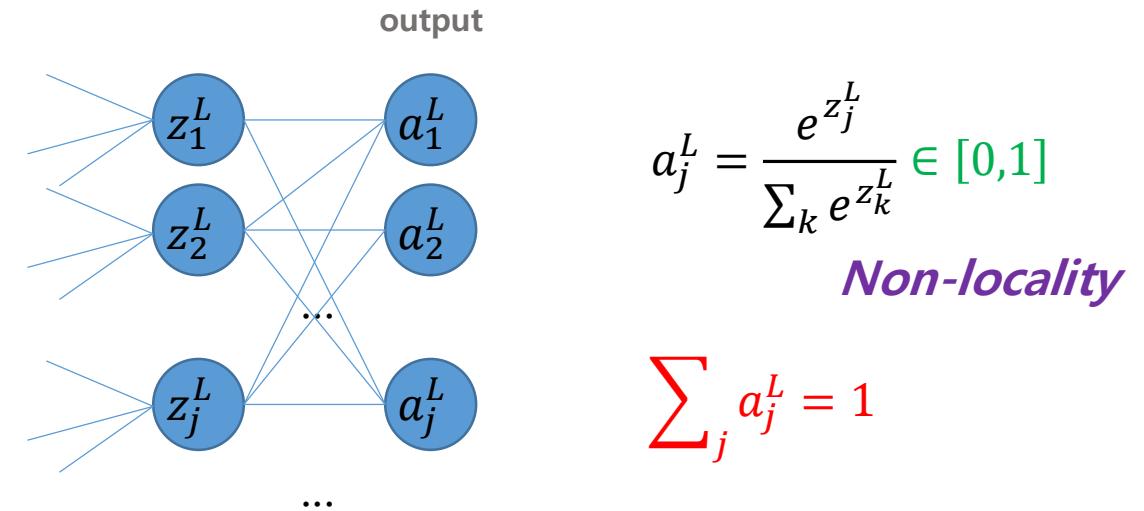
Original Output Layer



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



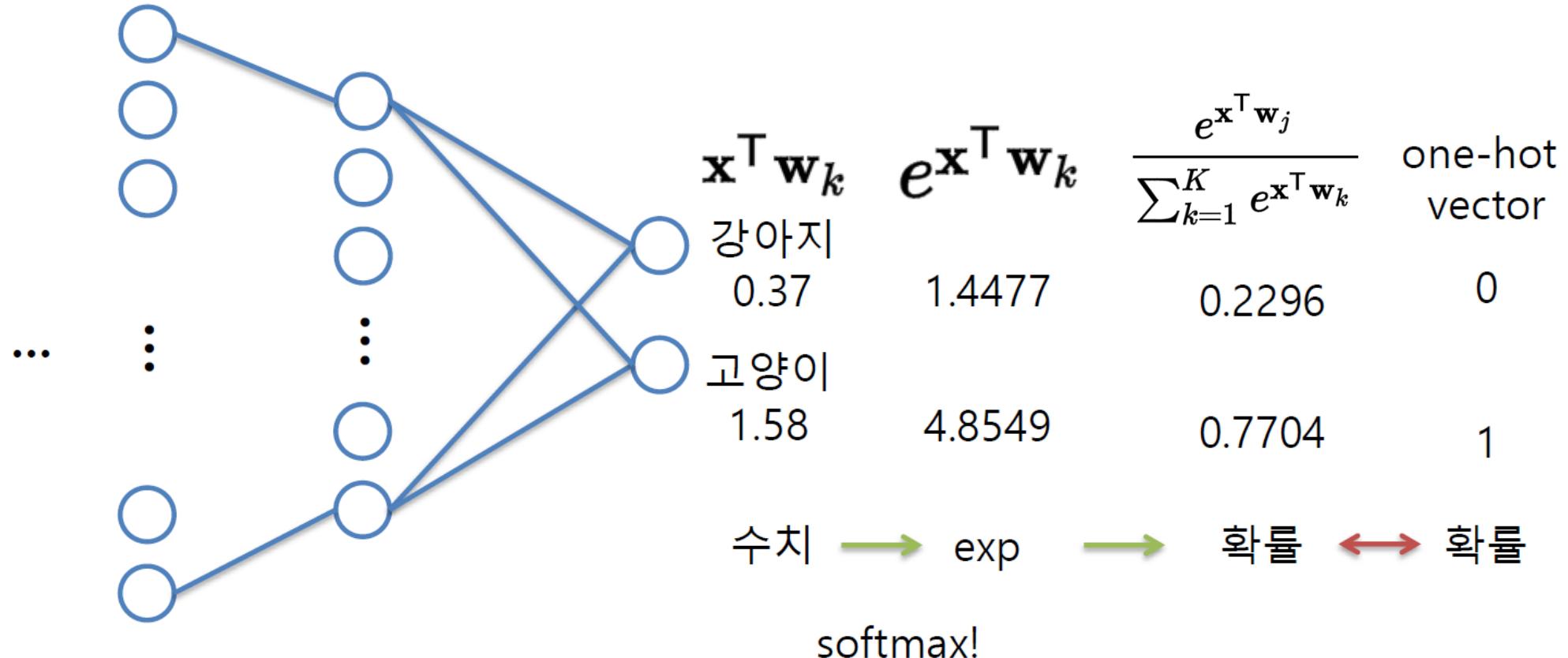
A softmax layer outputs a probability distribution!!

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

Softmax

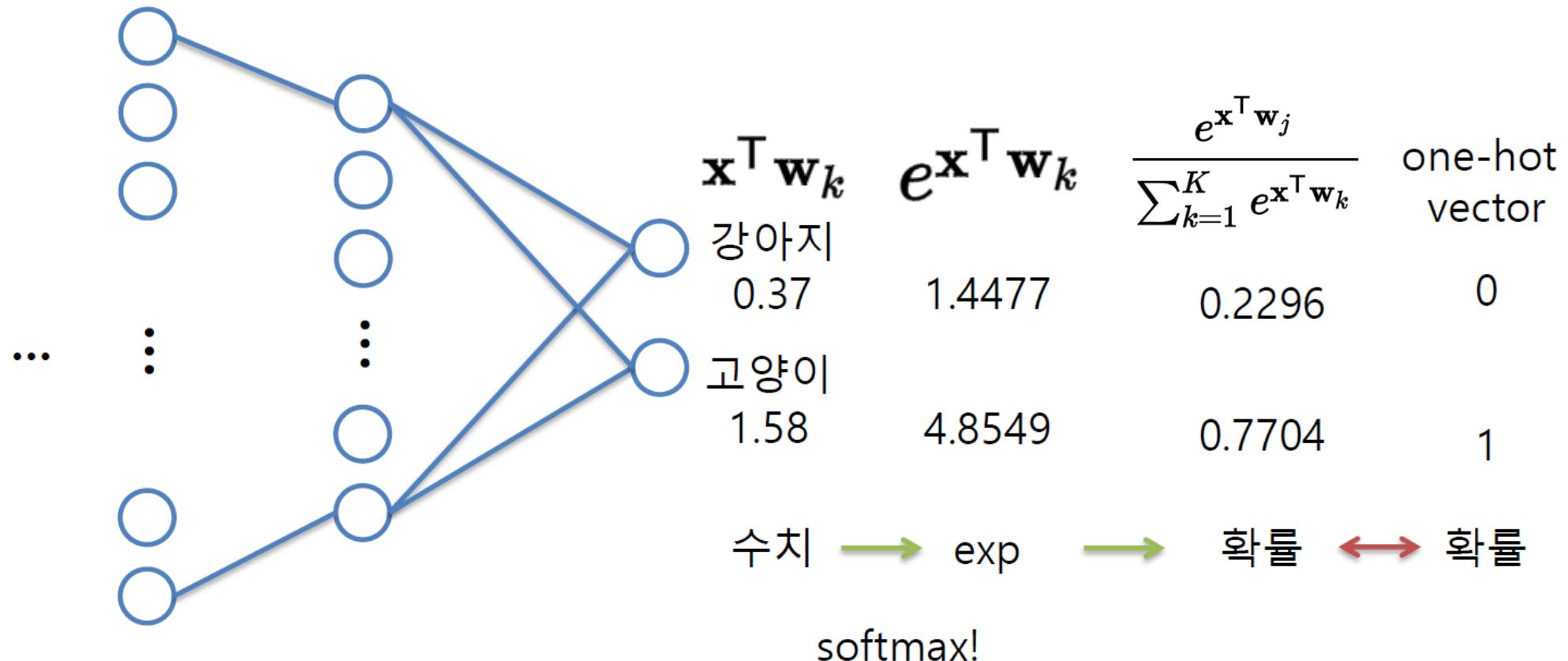
- Output을 확률처럼 나타내보자

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



Loss Function of Softmax

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



Loss Function of Logistic Regression

- Cross Entropy Loss!

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

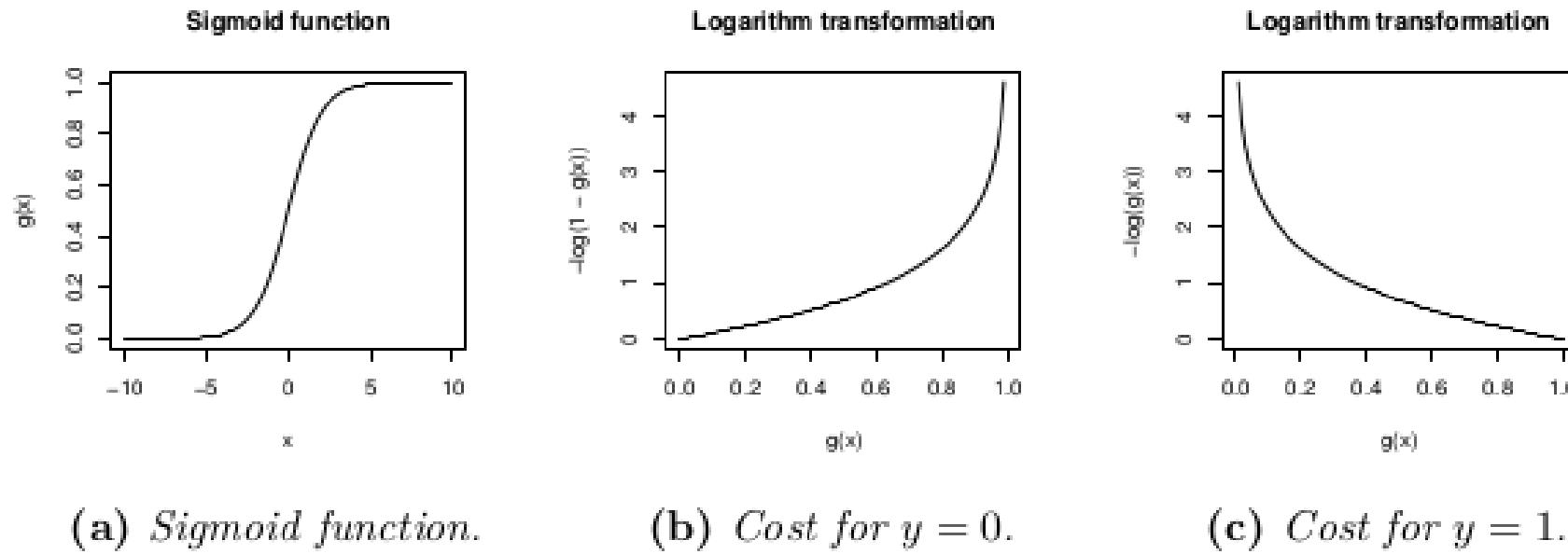
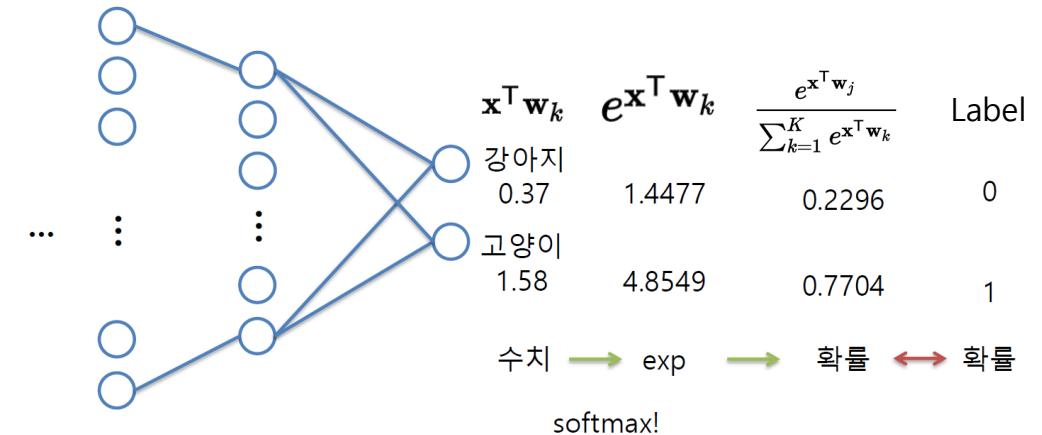


Figure B.1: Logarithmic transformation of the sigmoid function.

Loss Function – Classification

- 마지막 layer의 activation function

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

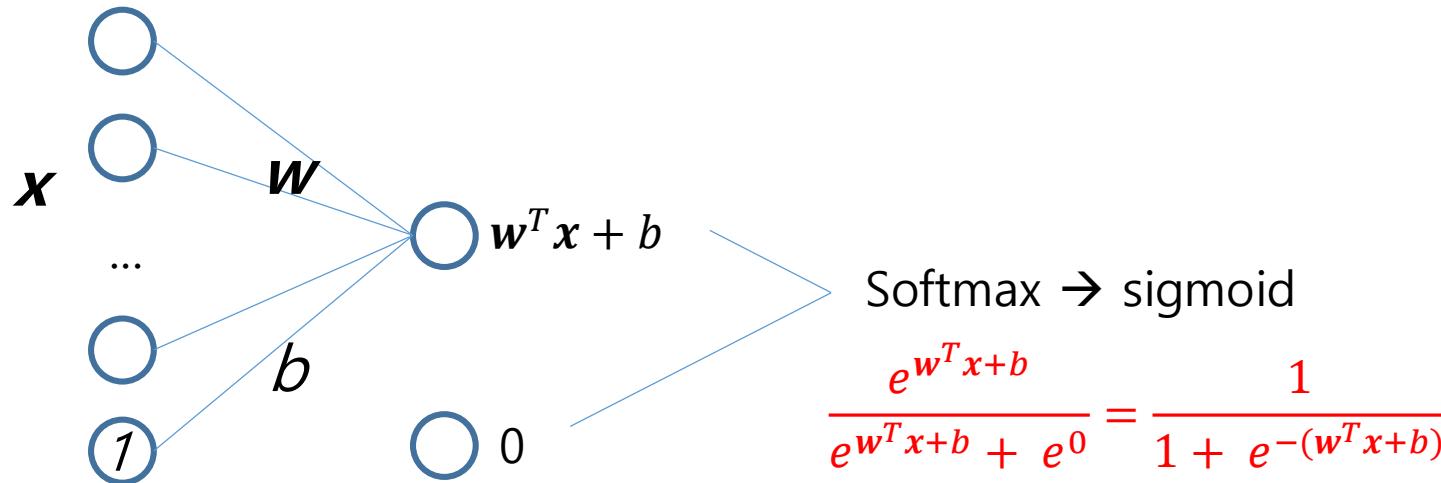


- Loss Function – Cross Entropy

- $L = -y \log H(x)$,
- y 는 label(정답), $H(x)$ 는 network output(예측값, softmax 결과)

Binary Classification

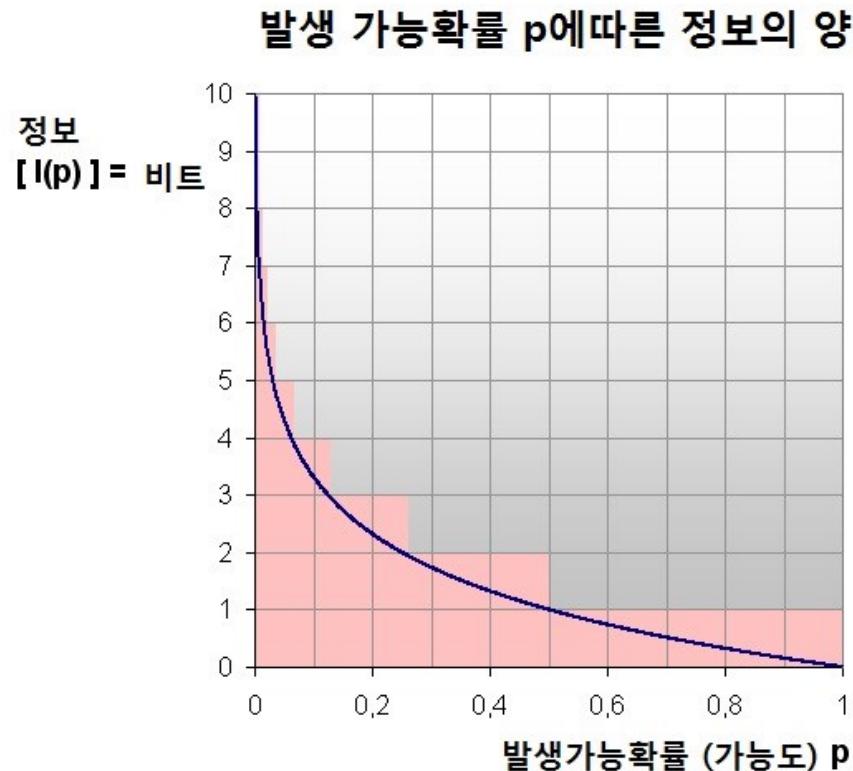
- 마지막 layer의 activation function
 - Sigmoid – $H(x) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}} = P(y|\mathbf{x})$



entropy

- Information Theory에서 Entropy란 정보량의 기댓값(평균)이다

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

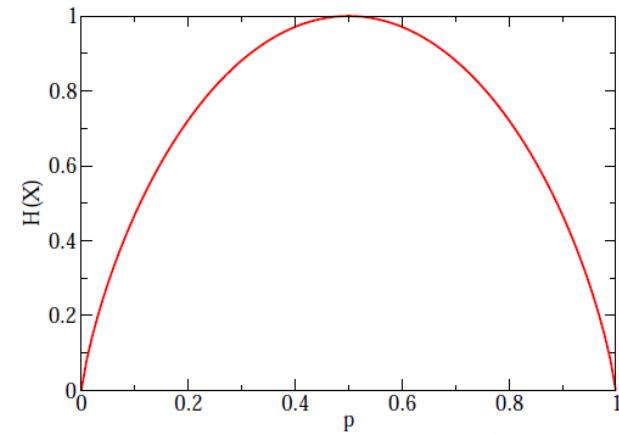


- 항상 일어나는 사건이라면 $p=1$ 이고 이것이 가지고 있는 정보량은 0이다.
- 일어날 확률이 적을 수록 많은 정보를 담고 있다.

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

$$P(\text{앞}) = 1/2$$

$$P(\text{뒤}) = 1/2$$

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

Cross Entropy

True distribution p,
Model이 예측한 distribution을 q라고 하면

$$H(p, q) = - \sum_i p_i \log q_i$$

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

$$= - \sum_i p_i \log q_i - \boxed{\sum_i p_i \log p_i} + \sum_i p_i \log p_i$$

$$= \boxed{H(p)} + \sum_i p_i \log p_i - \sum_i p_i \log q_i$$

$$= H(p) + \sum_i p_i \log \frac{p_i}{q_i}$$

$$= H(p) + D_{KL}(p \parallel q)$$

P 자체의 entropy(불변)
↑

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

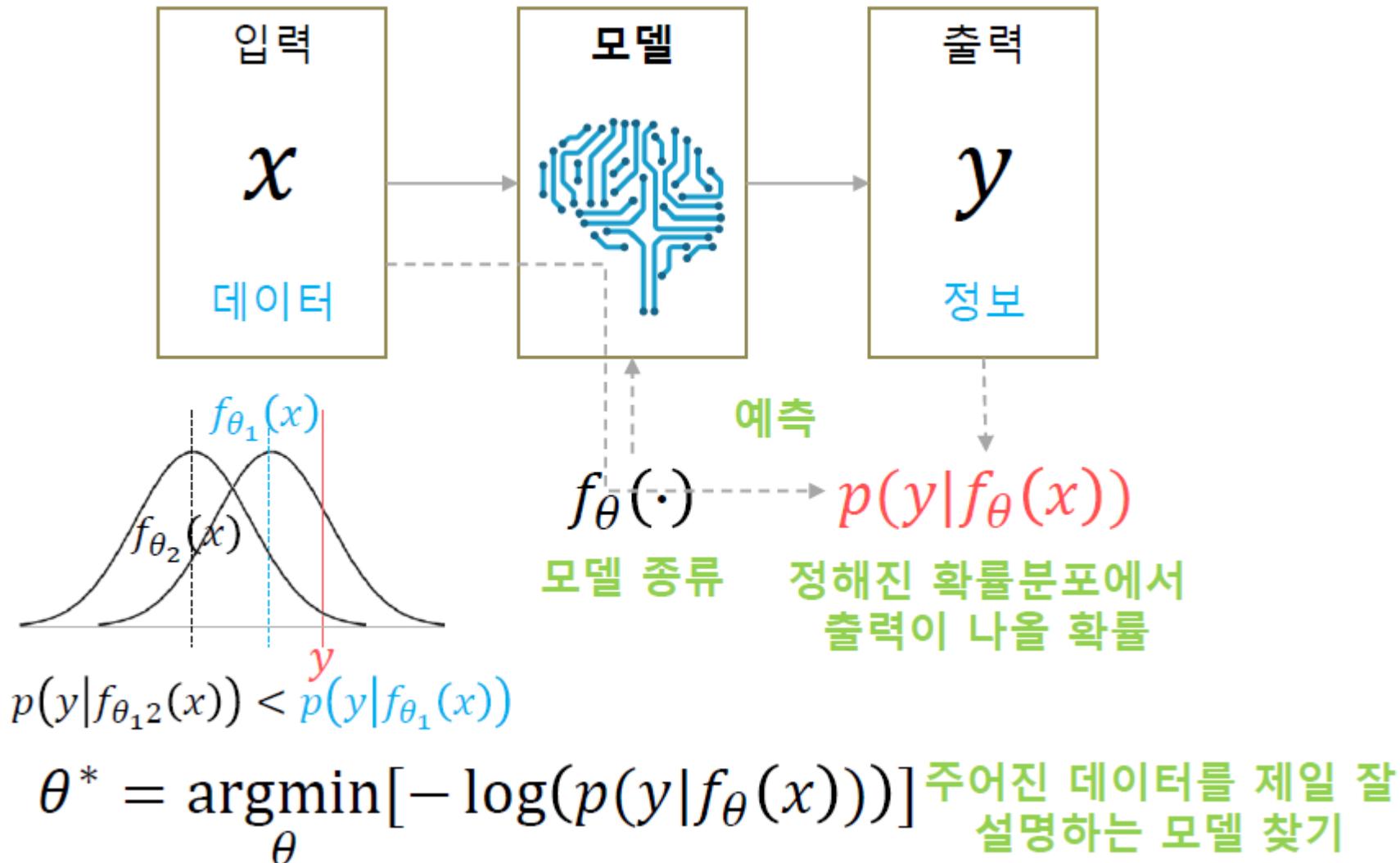
Kullback–Leibler(KL) Divergence

- 두 확률 분포간의 거리(?)를 나타냄
 - $D_{KL}(p \parallel q) = p \log \frac{p}{q} = p \log p + (-p \log q) = -H(p) + H(p, q) = H(p, q)$
 - p는 label, q는 network output 이라고 할 경우, network의 목표는 label이 나 타내는 확률 분포 p와 최대한 가까운 q를 학습하고자 하는 것임
 - 일반적으로 label은 one-hot encoding 하기 때문에, 각 label의 값이 확률이 라고 생각하면 p의 entropy, $H(p) = 0$
 - $D_{KL}(p \parallel q) > 0$

p의 entropy

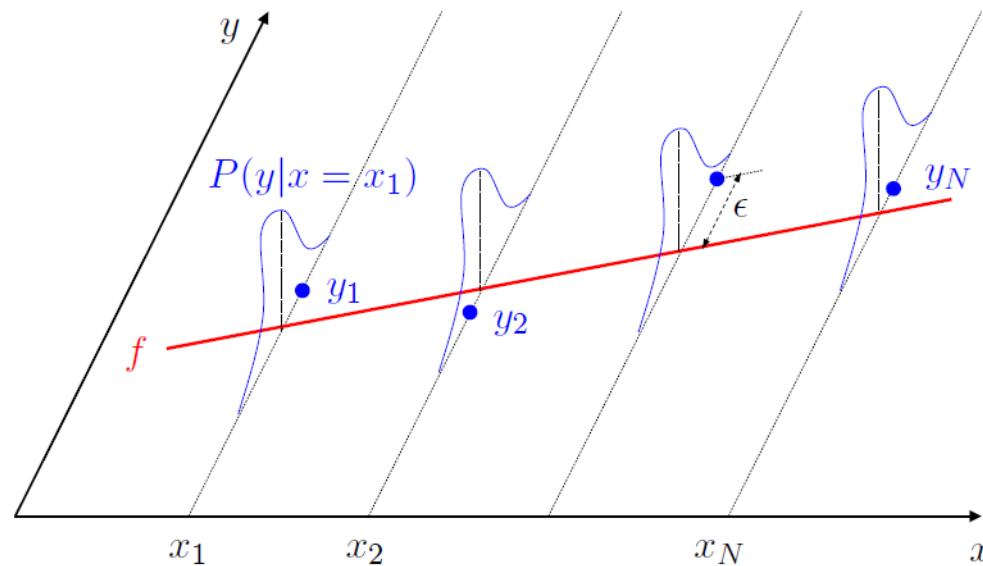
p, q의 cross entropy

Probabilistic View



Probabilistic View

- Linear Regression
 - Training data is considered as the set of iid samples from underlying probabilistic model:
$$y_i = \theta^T x_i + \epsilon_i, \text{ where } \epsilon_i \text{ follows Gaussian distribution } \epsilon_i \sim N(0, \sigma^2)$$



- Models predict the mean of Gaussian distribution(μ_i)

Probabilistic View

- Logistic Regression
 - Training data is considered as the set of iid samples which follows Bernoulli distribution:
 - Models predict the probability $p(y = 1|x, \theta)$ directly

$$p(x) = p^x(1 - p)^{(1-x)}$$

Probabilistic View

– Minimize Negative Log-Likelihood

Univariate cases

$$-\log(p(y_i|f_\theta(x_i)))$$

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \sigma_i = 1$$

$$p(y_i|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}\right)$$

$$\log(p(y_i|\mu_i, \sigma_i)) = \log\frac{1}{\sqrt{2\pi}\sigma_i} - \frac{(y_i - \mu_i)^2}{2\sigma_i^2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{\sqrt{2\pi}} + \frac{(y_i - \mu_i)^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{(y_i - \mu_i)^2}{2} = \frac{(y_i - f_\theta(x_i))^2}{2}$$

Mean Squared Error

Bernoulli distribution

$$f_\theta(x_i) = p_i$$

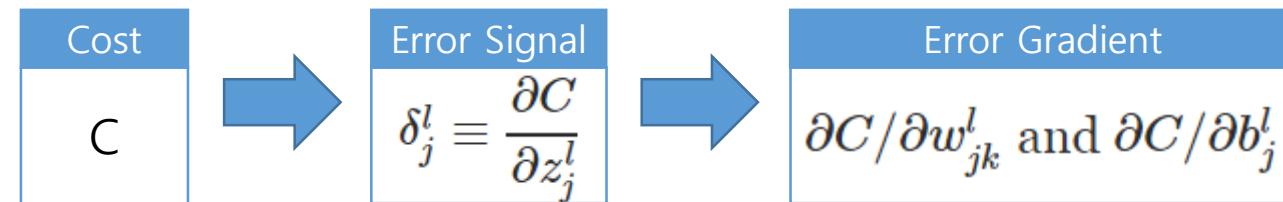
$$p(y_i|p_i) = p_i^{y_i} (1 - p_i)^{1-y_i}$$

$$\log(p(y_i|p_i)) = y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

$$-\log(p(y_i|p_i)) = -[y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

Cross-entropy

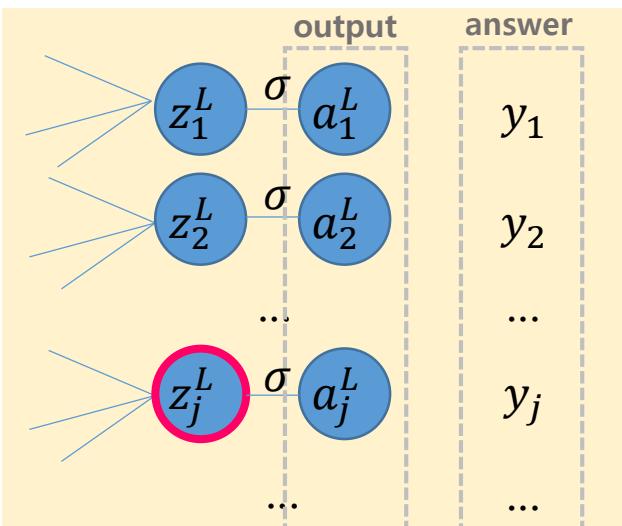
Fundamental Equation behind Back-Prop



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

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

Chain rule



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

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

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

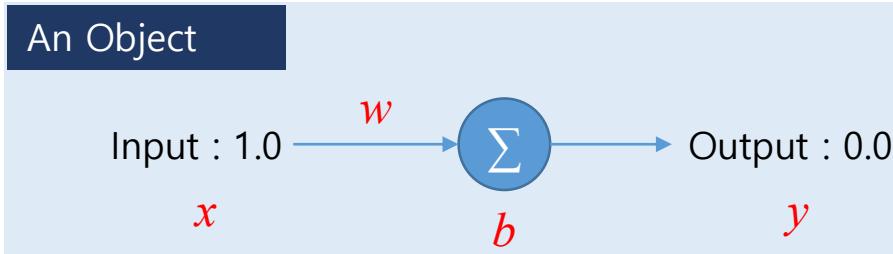
Only the j^{th} term is valid

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

Mean Squared Error

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

An example :



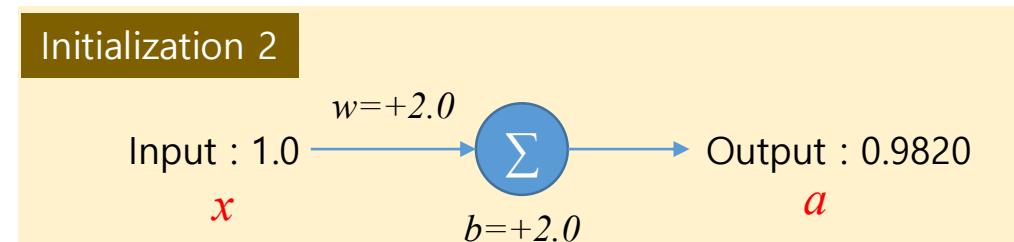
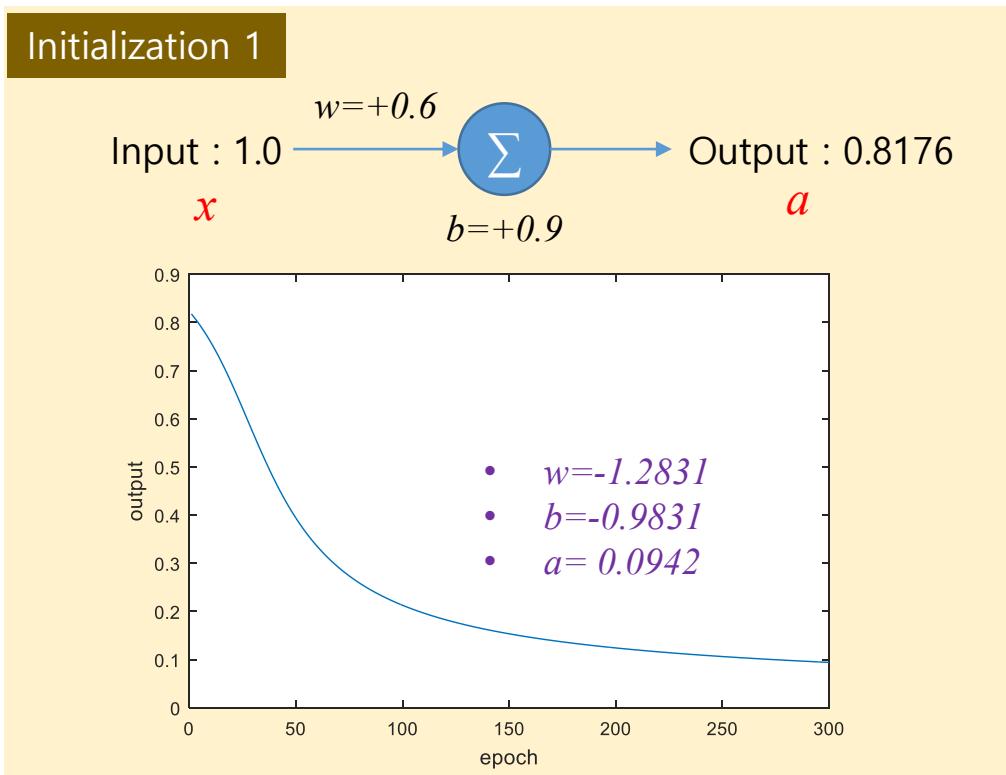
$$C = \frac{(a - y)^2}{2} = \frac{a^2}{2}$$

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

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

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

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

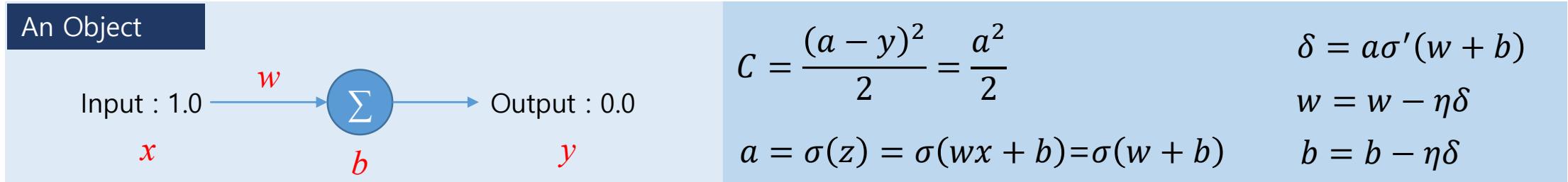


difficulty learning when it's badly wrong

Mean Squared Error

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

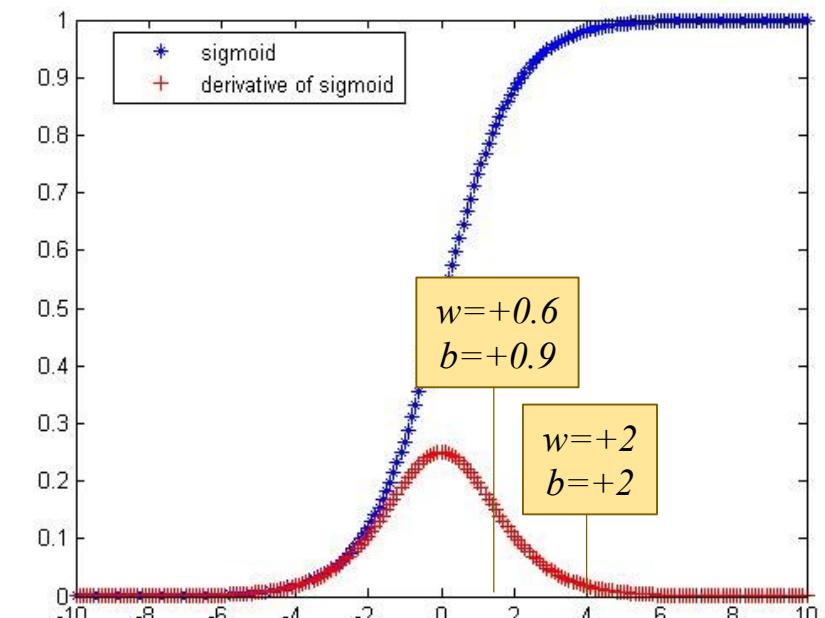
An example :



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

Why they are small???

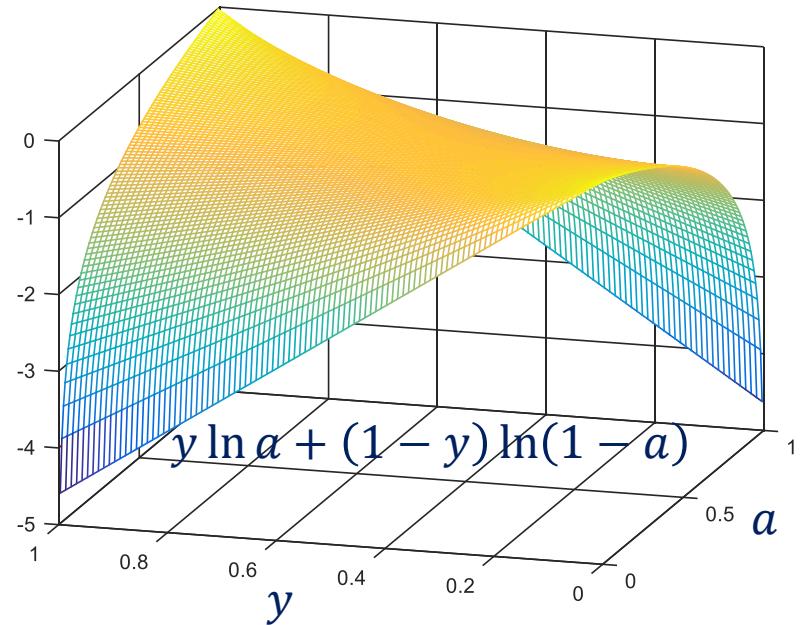
$$C = \frac{(y - a)^2}{2}$$
$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) = a\sigma'(z)$$



Cross Entropy Cost Function

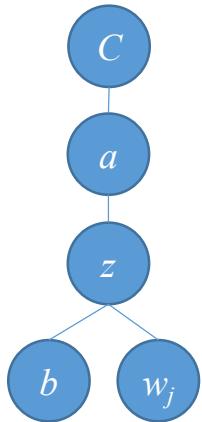
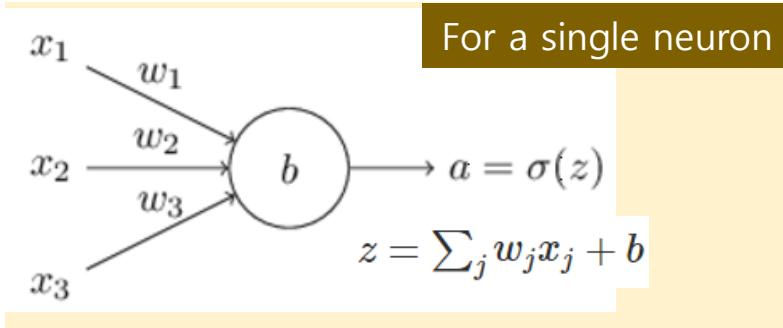
Introducing the cross-entropy cost function

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



[Two features]

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



Error gradient

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

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

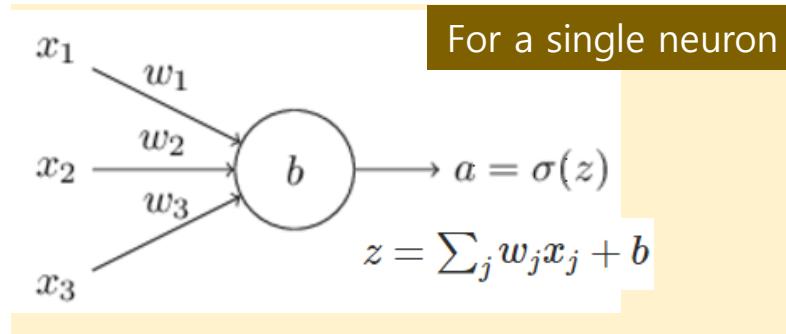
$$\frac{\partial a}{\partial z} = \sigma'(z) = (1 - \sigma(z))\sigma(z) = (1 - a)a \quad \frac{\partial z}{\partial w_j} = x_j$$

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

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

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!!!

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