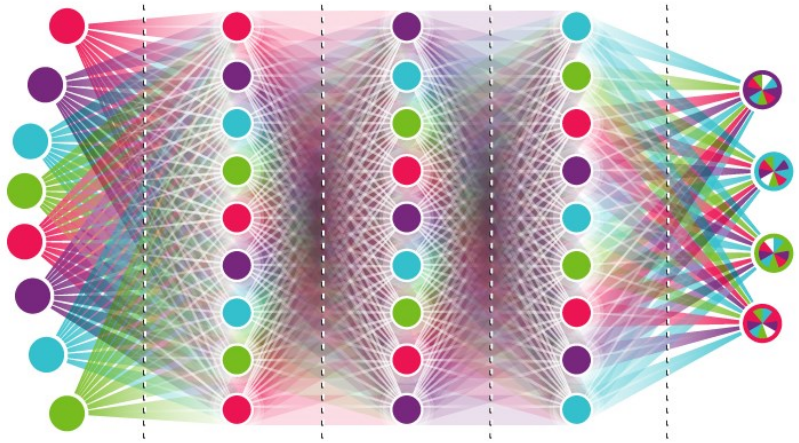
Multi-Layer Perceptron

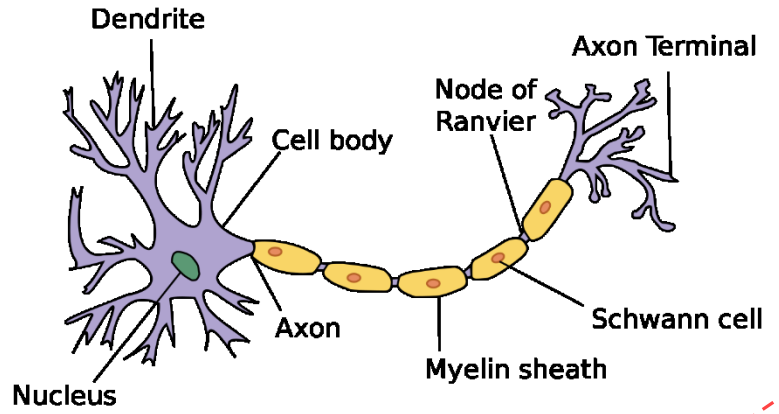
The Road to Deep Learning

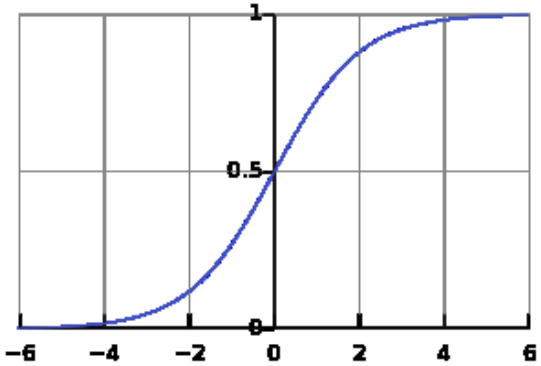


Fast Campus

Start Deep Learning with Tensorflow

Recap - Perceptron





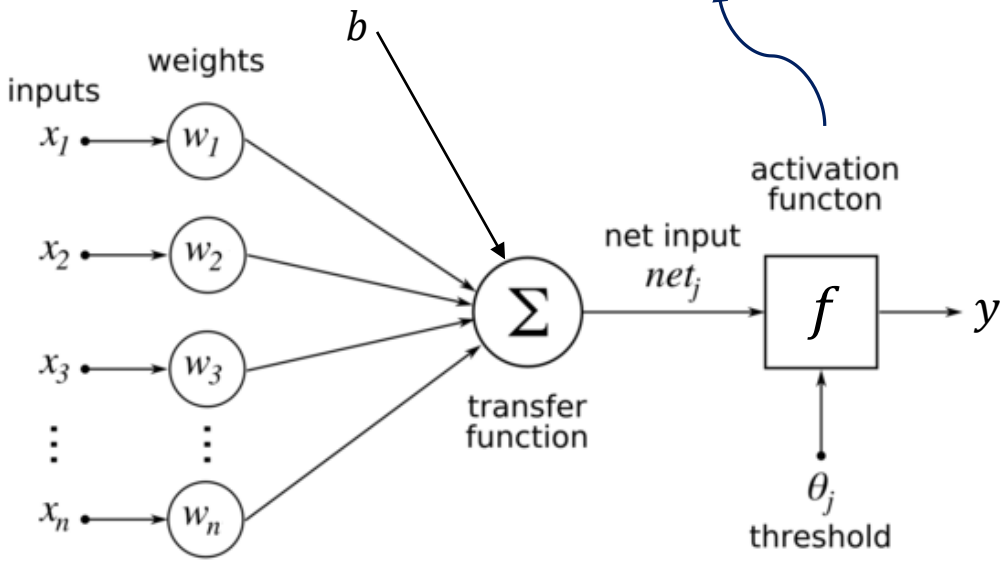
sigmoid activation function

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

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

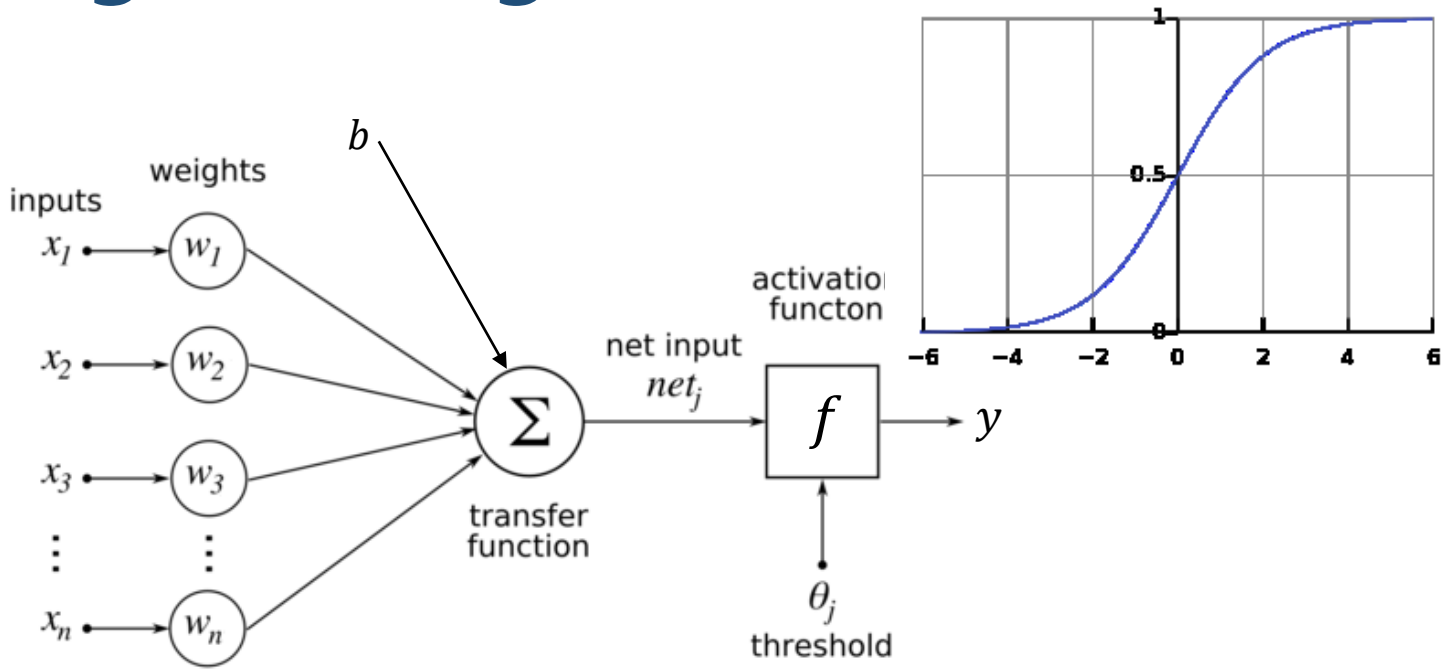
$$\boldsymbol{w} = [w_1\ w_2\ w_3\ \ldots\ w_n]$$

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



<Perceptron>

Recap – Logistic Regression



<Perceptron>

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases} \xrightarrow{\text{sigmoid activation}} \text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

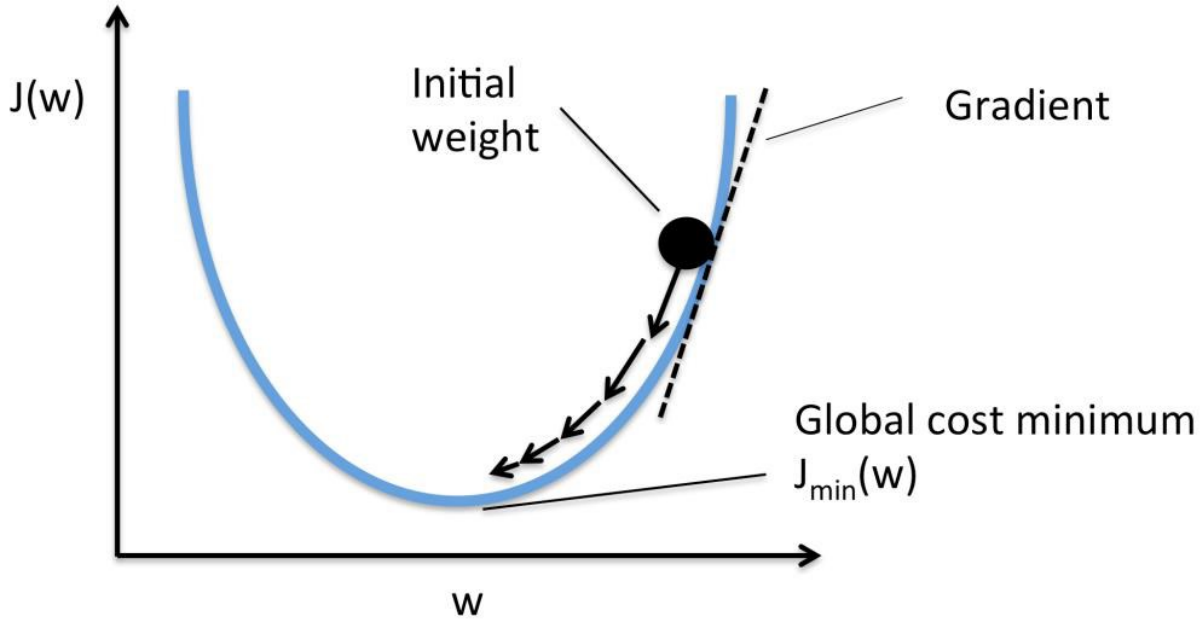
Sigmoid activation : small changes in their weights and bias cause only a small change in their output

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

Recap – Training(Gradient Descent)

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

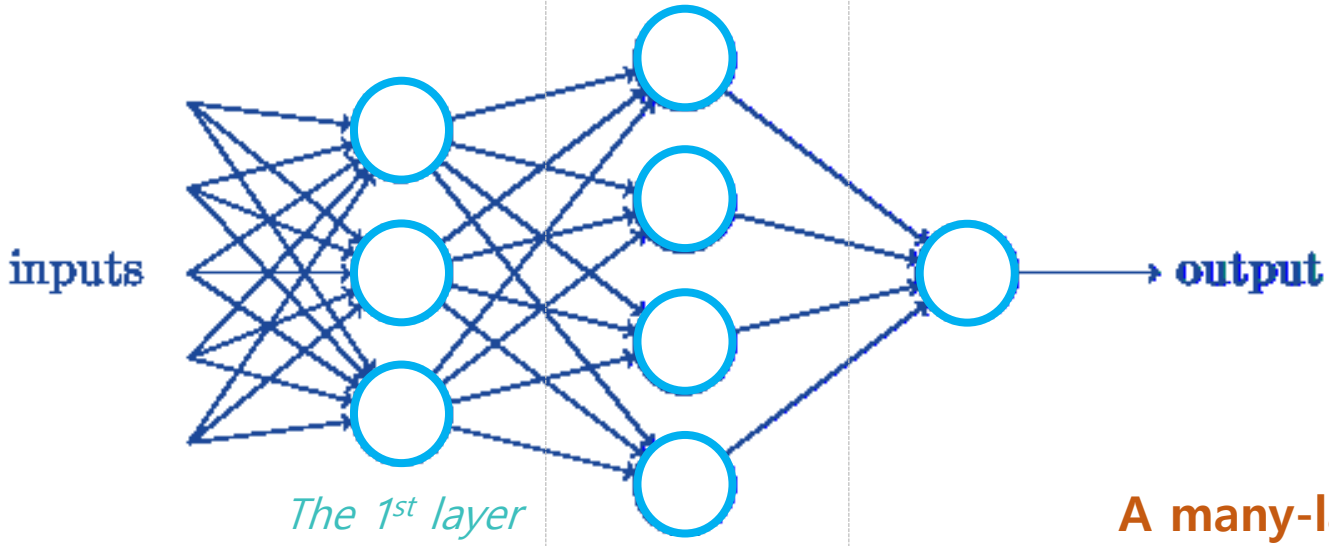
- 방향 : 그 지점에서의 – gradient
- 속력(보폭) : learning rate(α)



Linear Regression vs Logistic Regression

Type	Pros	Cons
Linear Regression	• Simple to implement	• Not guaranteed to work • Only supports binary labels
Logistic Regression	• Highly accurate • Model responses are measures of probability	• Only supports binary labels

How about This? Multi-Layer Perceptron



The 1st layer

simple decisions are made

The 2nd layer

more complex and more complex level decisions are made

The 3rd layer

Most complex decision is made

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

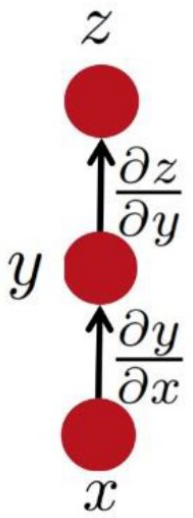
Network을 deep하게 쌓고, class도 여러 개일 때는
어떻게 학습할 수 있을까?

먼저 Multi-Layer부터 생각해봅시다

Back Propagation!

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

Simple Chain Rule

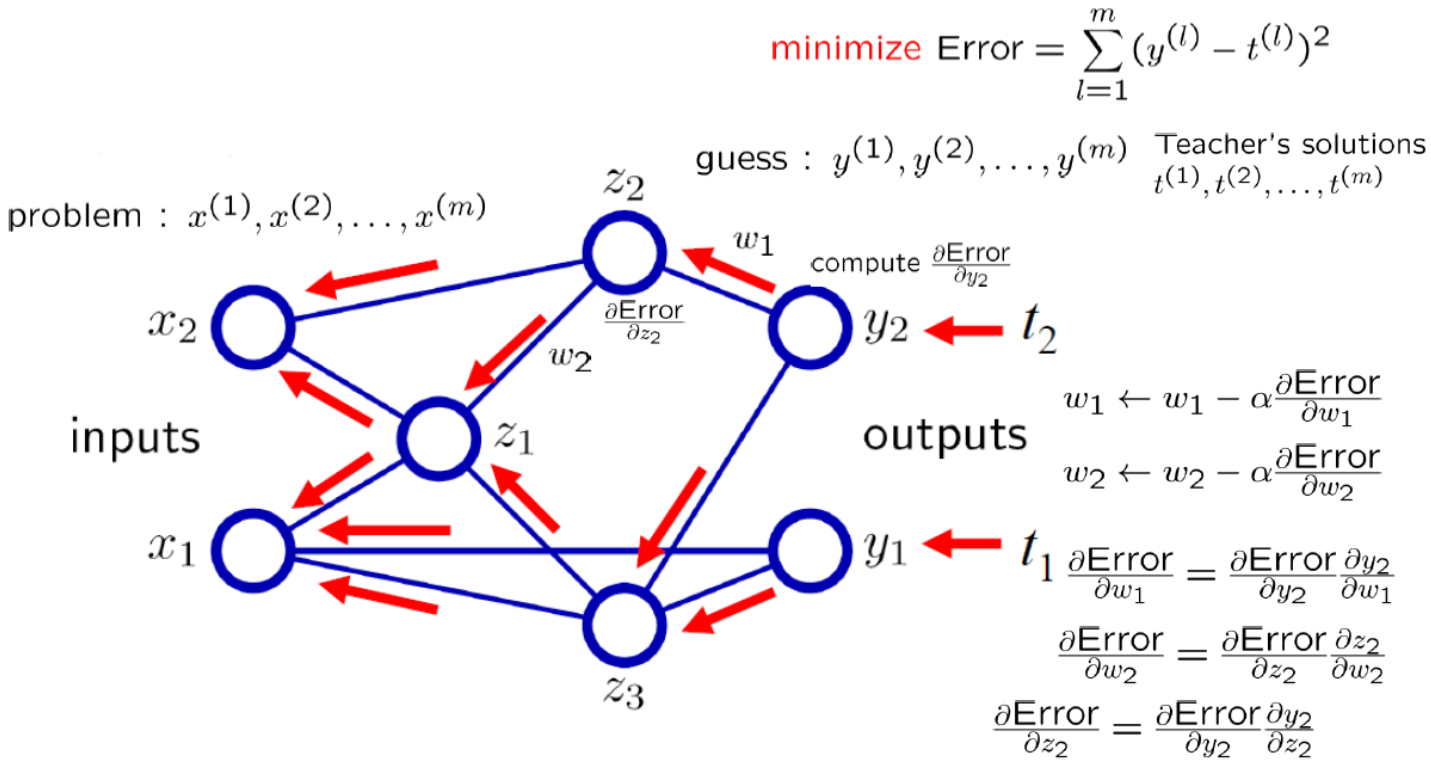


$$\Delta z = \frac{\partial z}{\partial y}\Delta y$$

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

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

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

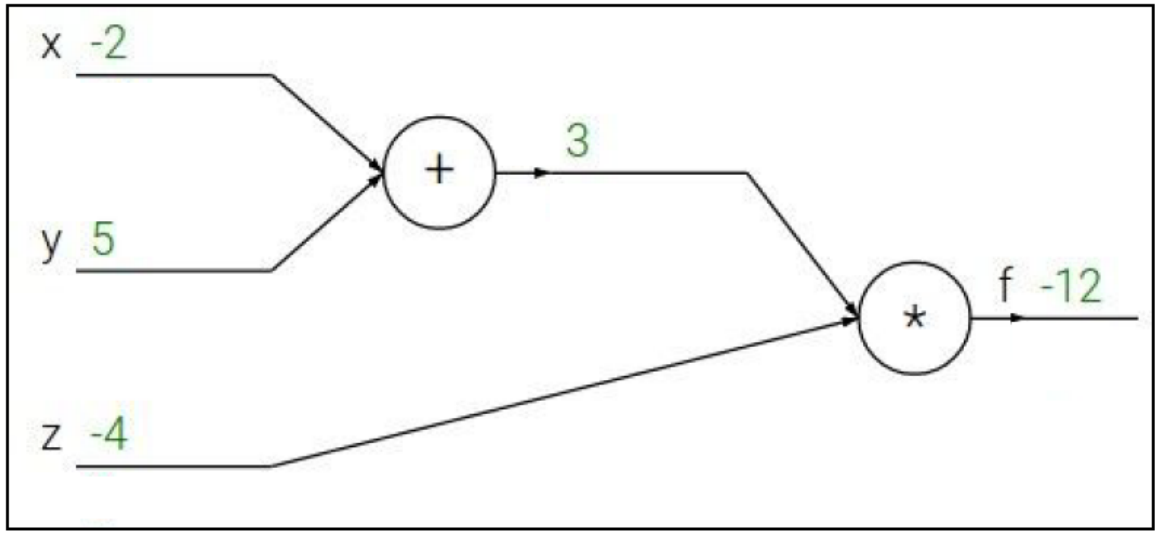


Back Propagation

Backpropagation: a simple example

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

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



Back Propagation

Backpropagation: a simple example

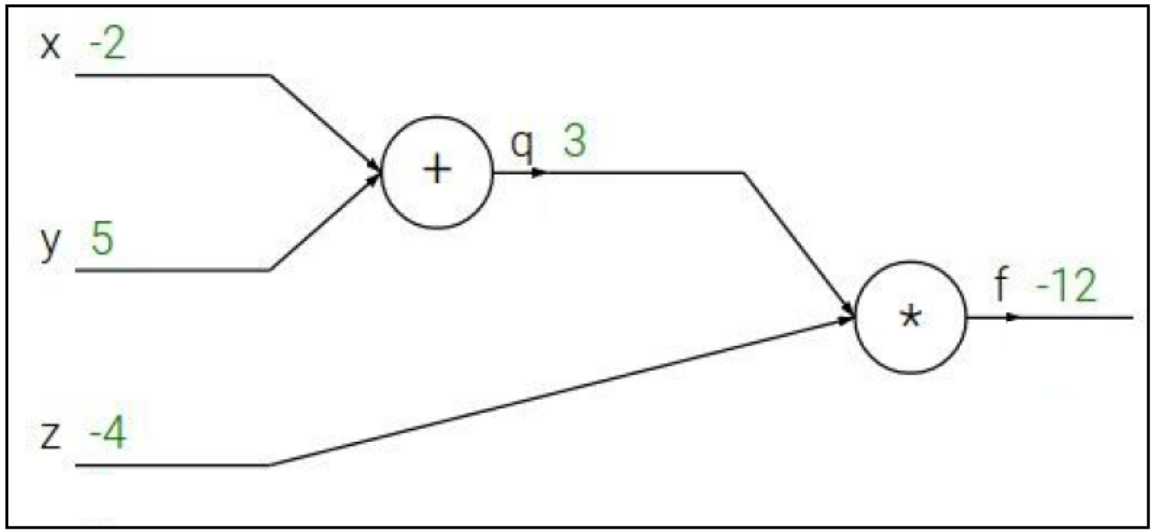
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Back Propagation

Backpropagation: a simple example

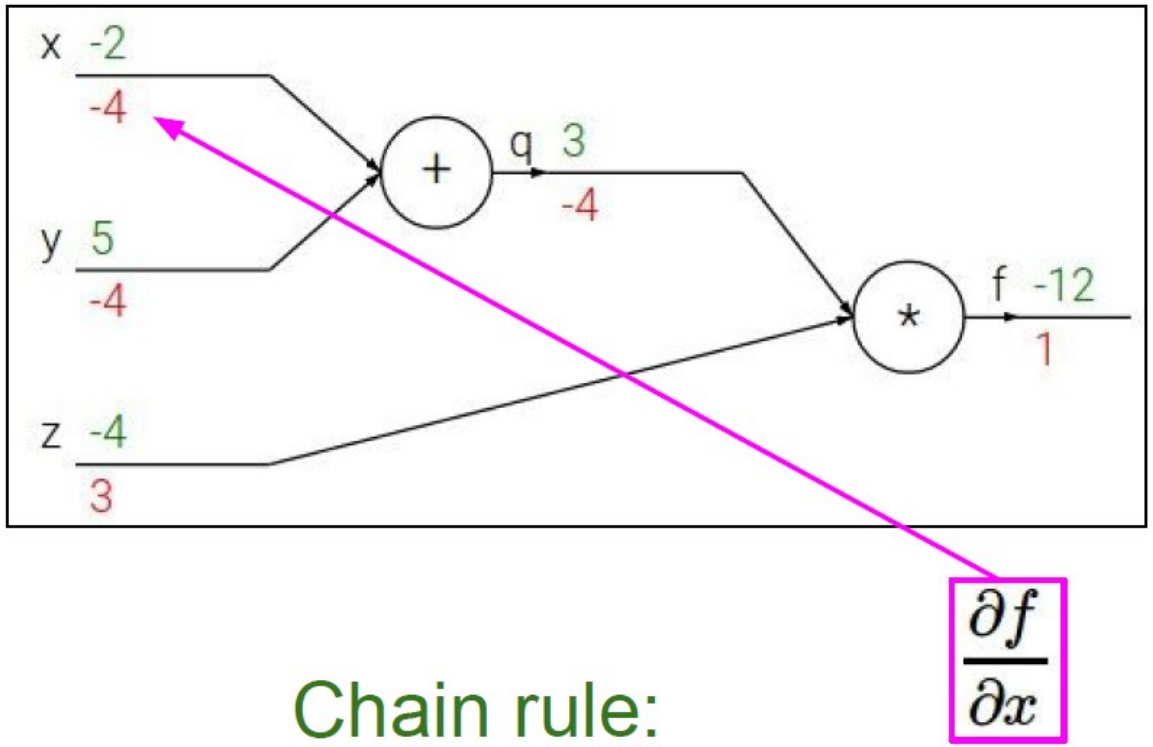
$$f(x, y, z) = (x + y)z$$

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

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

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

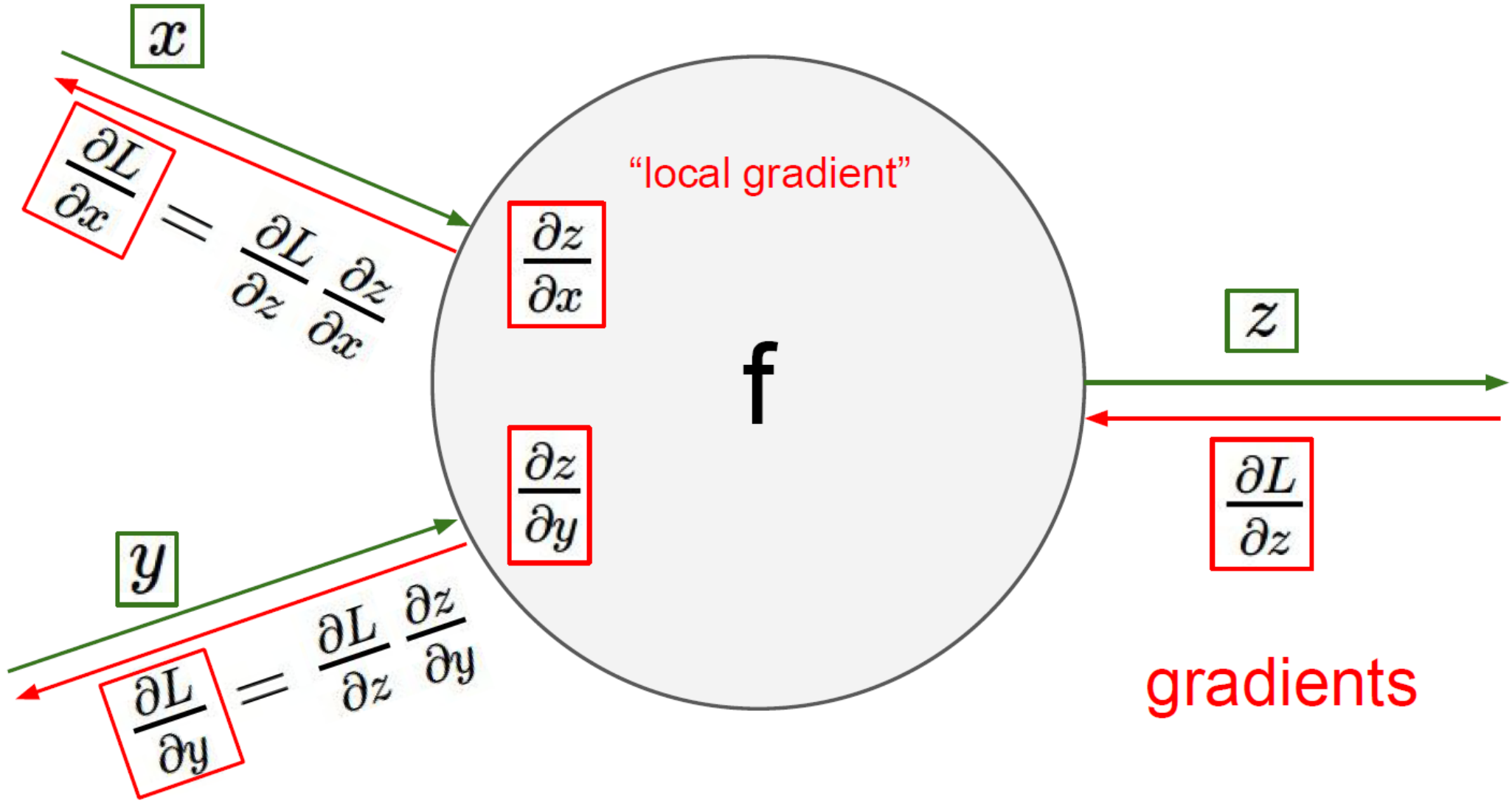
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

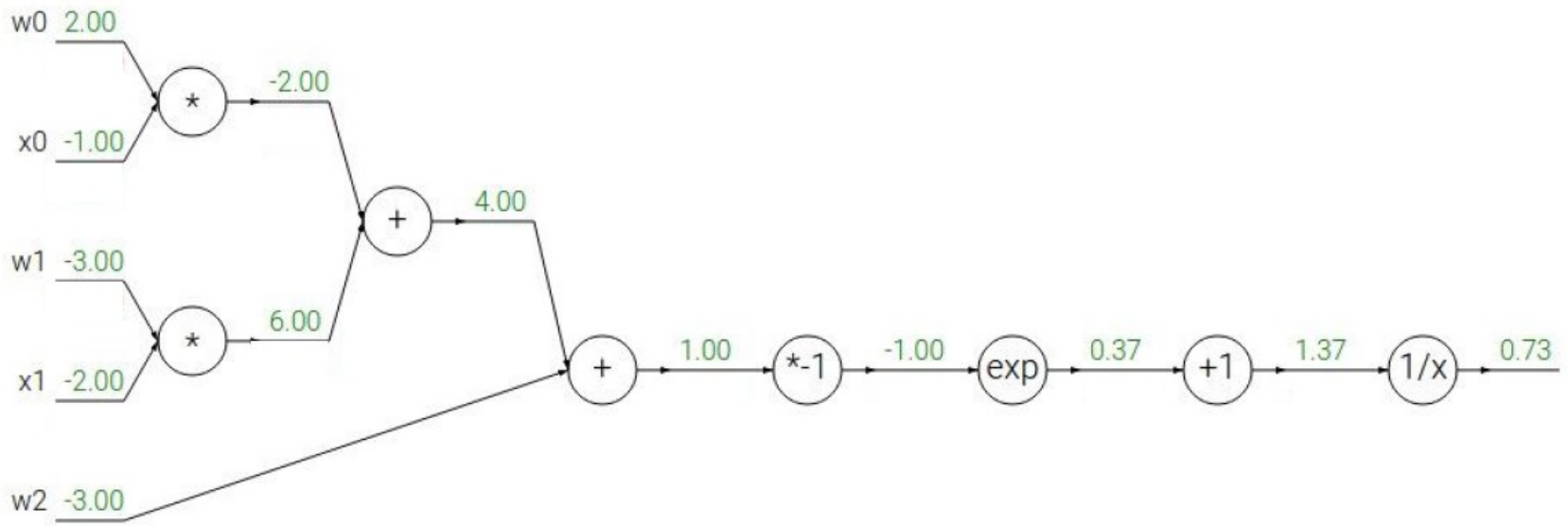
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Chain Rule(Local Gradient)



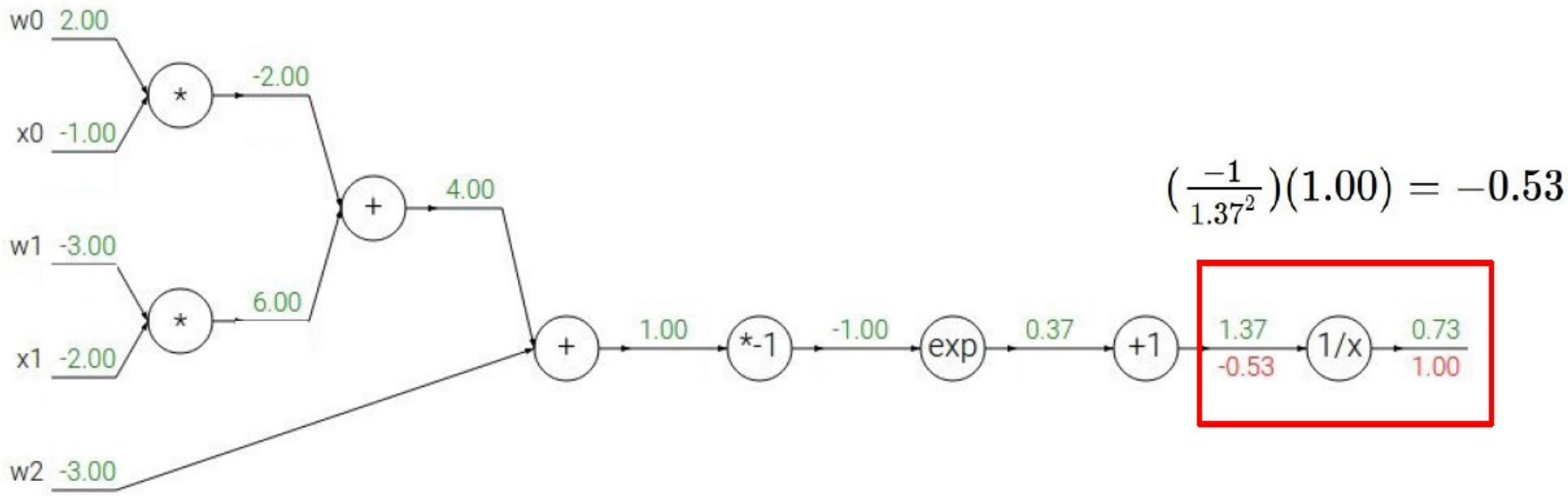
Back Propagation(Example)

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



Back Propagation(Example)

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



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

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

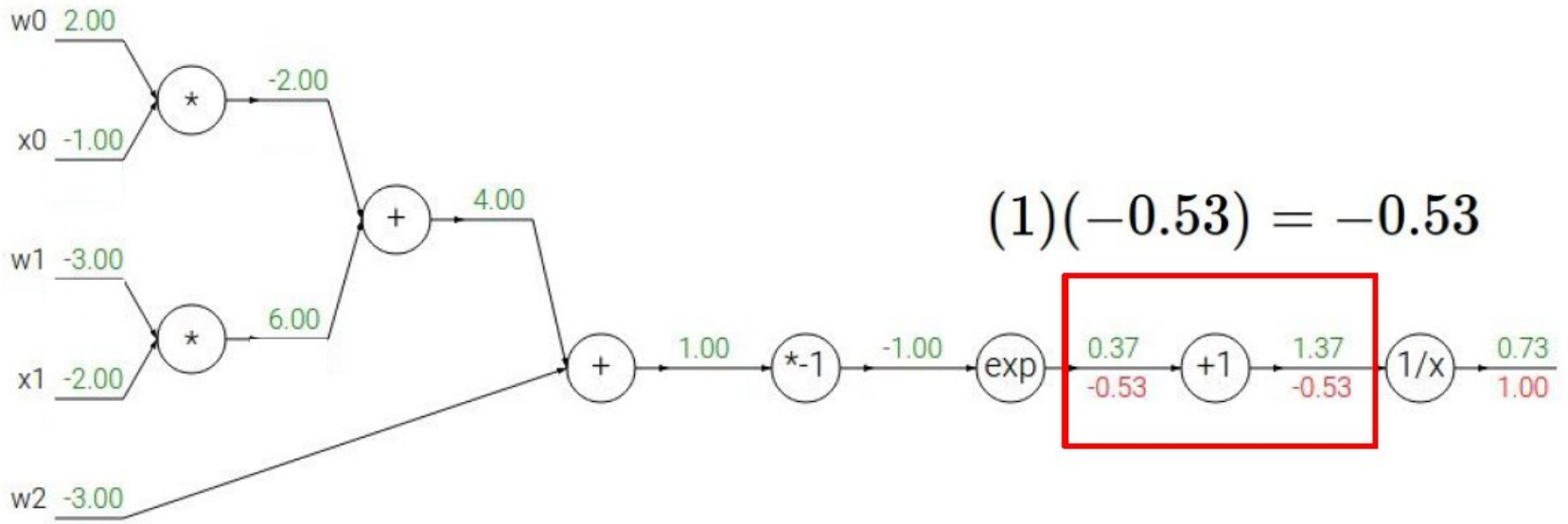
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Back Propagation(Example)

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



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

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

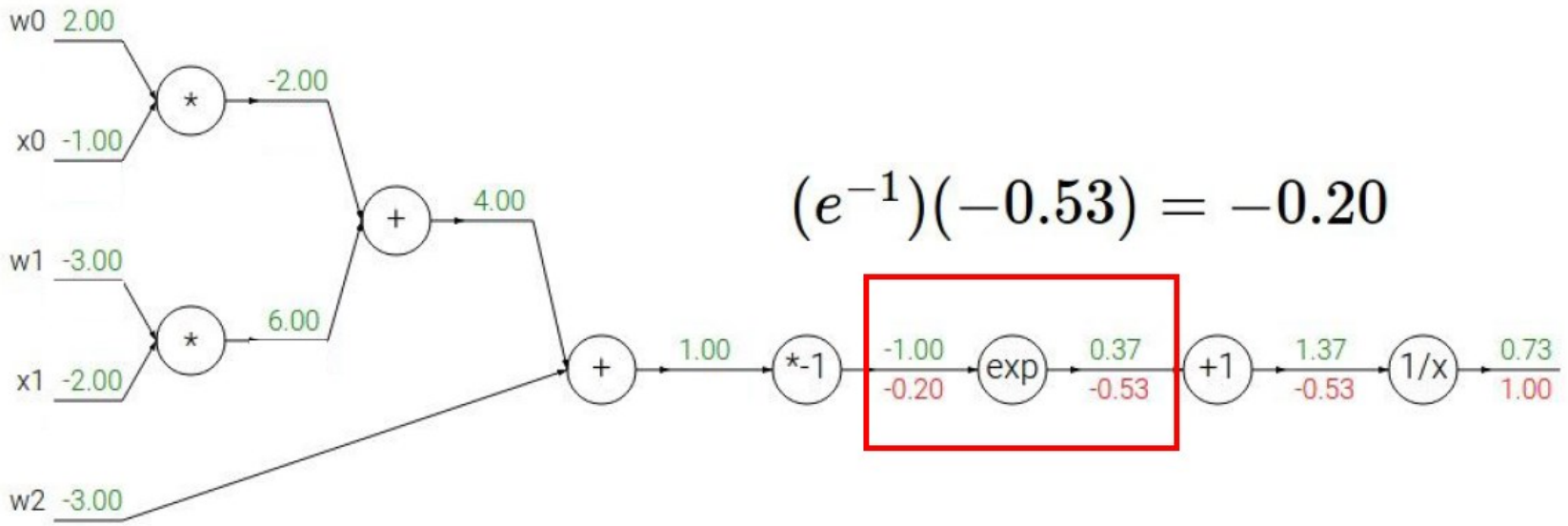
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Back Propagation(Example)

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



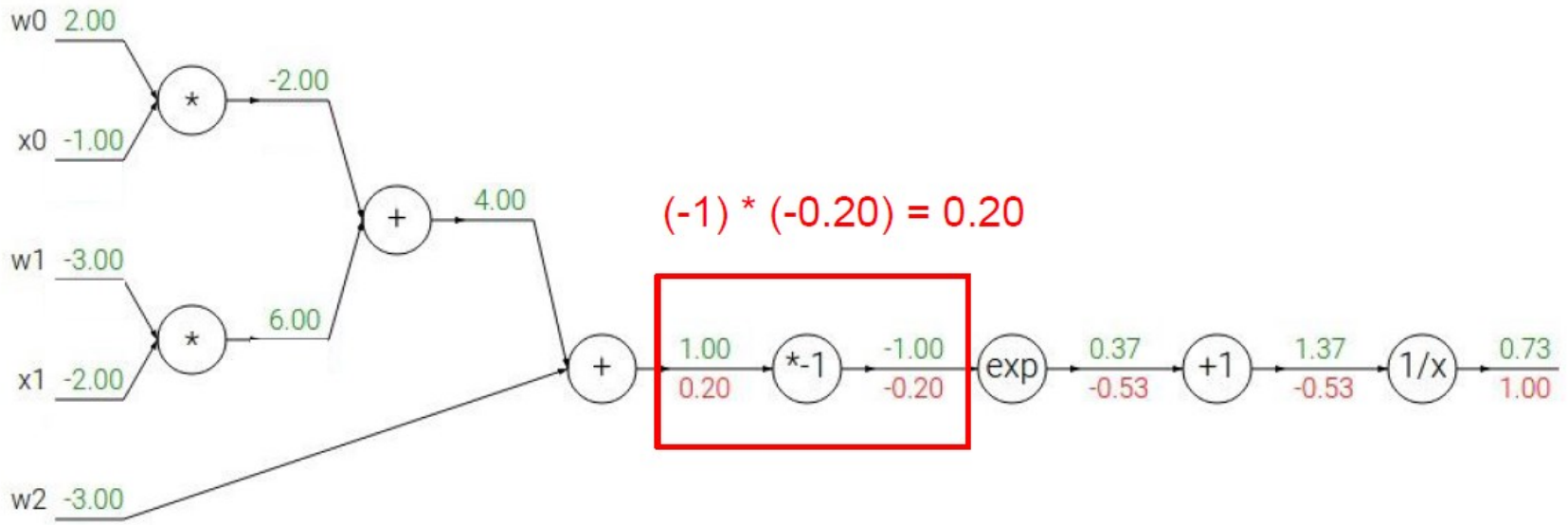
$$(e^{-1})(-0.53) = -0.20$$

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

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

Back Propagation(Example)

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



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

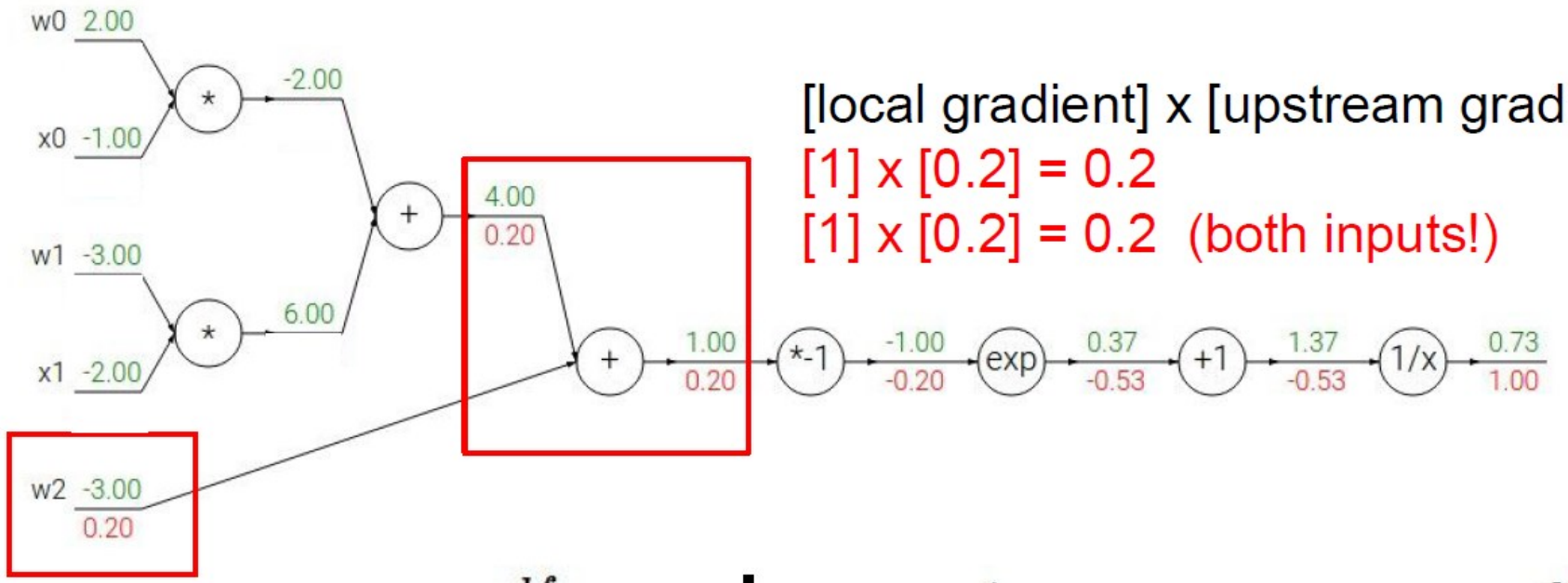
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Back Propagation(Example)

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



[local gradient] x [upstream gradient]

[1] x [0.2] = 0.2

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

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

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

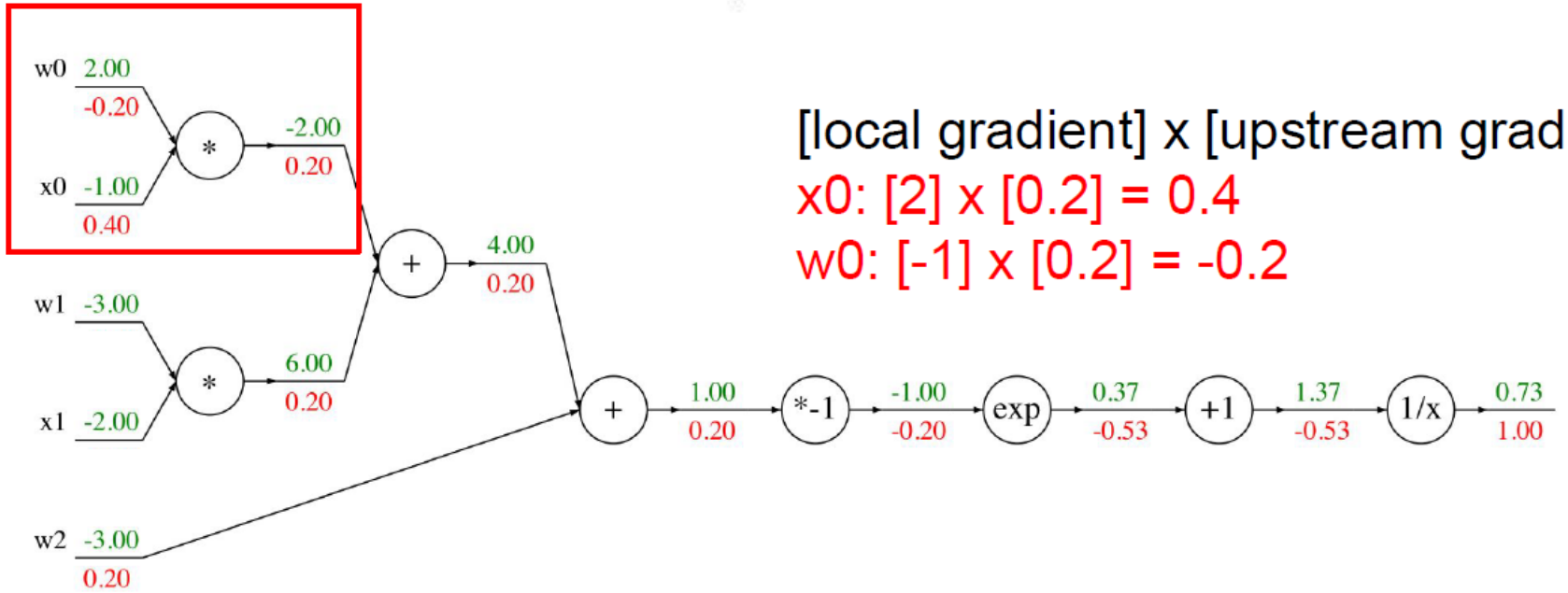
$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

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

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

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

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

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

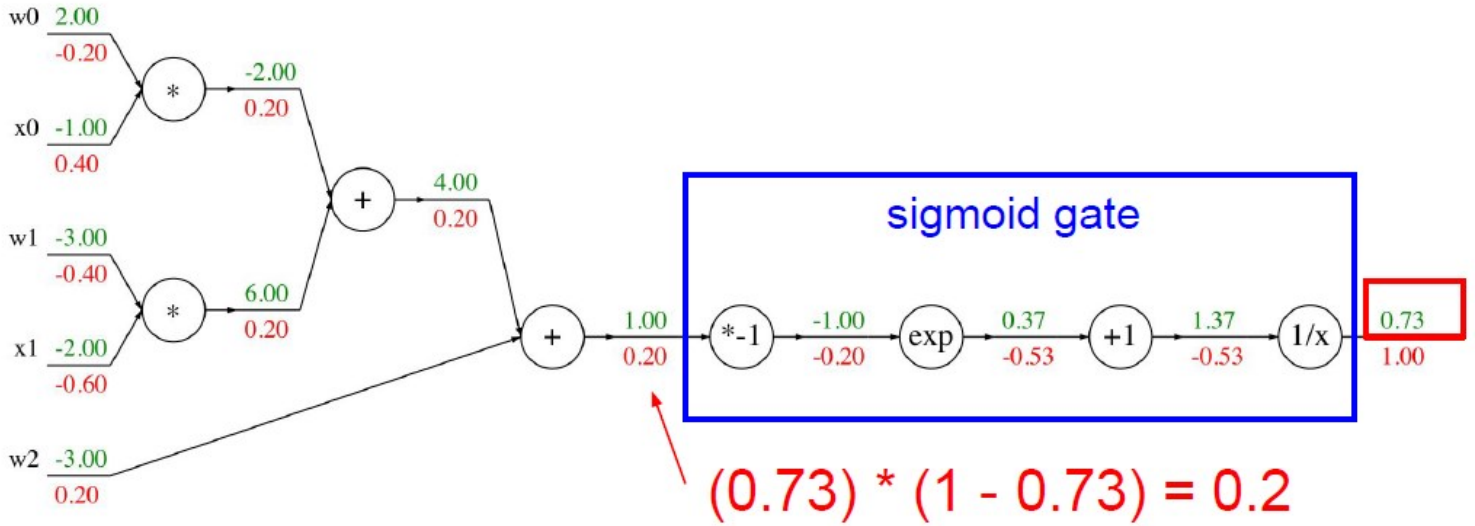
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Back Propagation(Example)

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

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

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



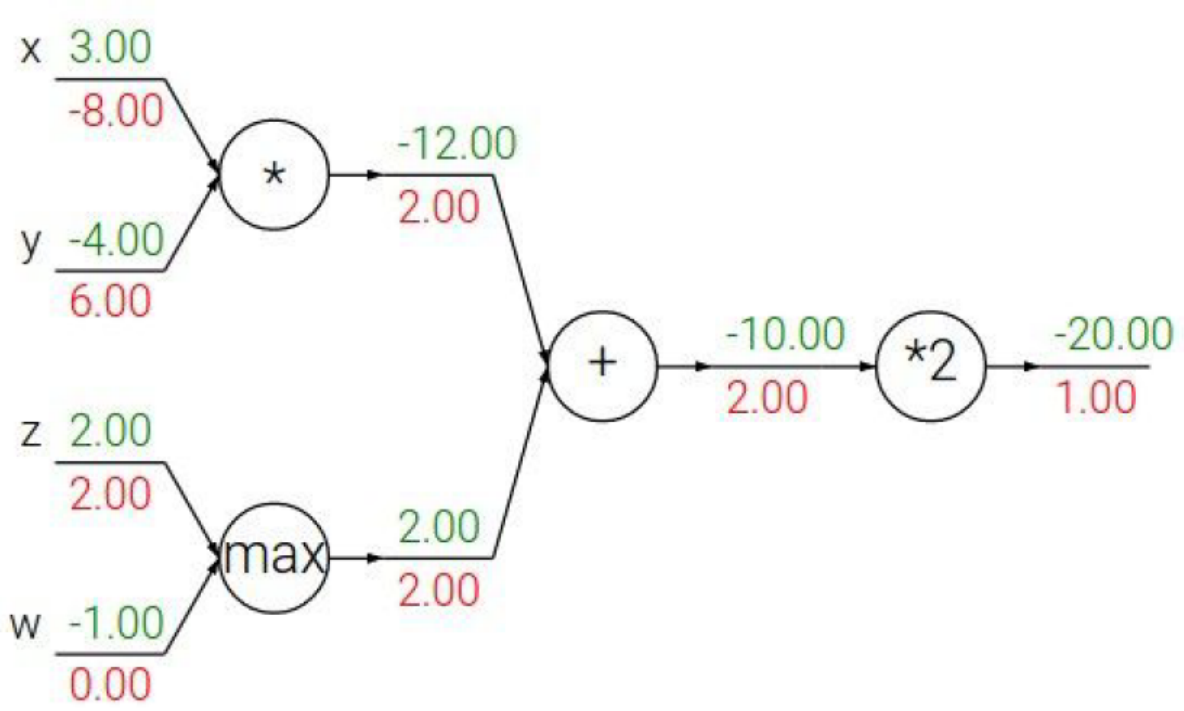
Back Propagation(Example)

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

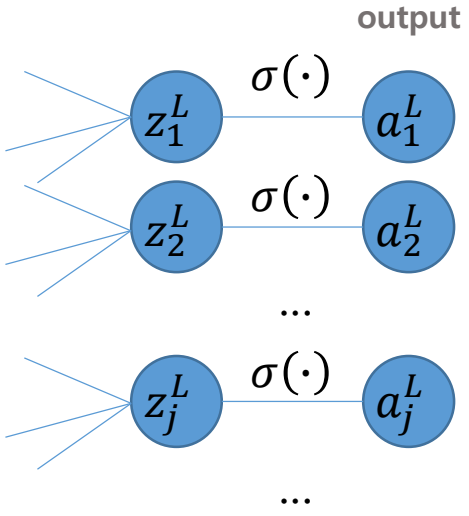
mul gate: gradient switcher



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

Softmax!

Original Output Layer



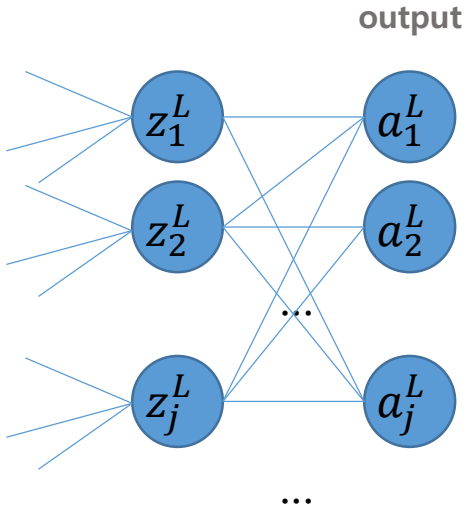
$$a_j^L = \sigma(z_j^L) = \frac{1}{1+e^{-z_j^L}} \in [0,1]$$

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

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

$$\sum_j a_j^L = 1$$

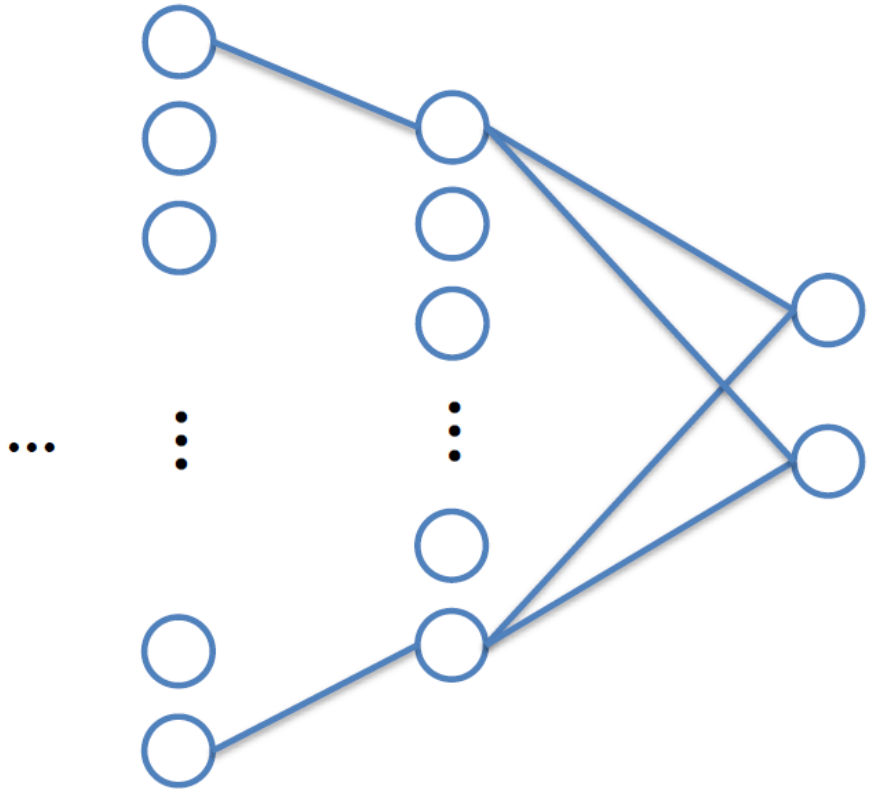
A softmax layer outputs a probability distribution!!

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

Softmax

- Output을 확률처럼 나타내보자

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



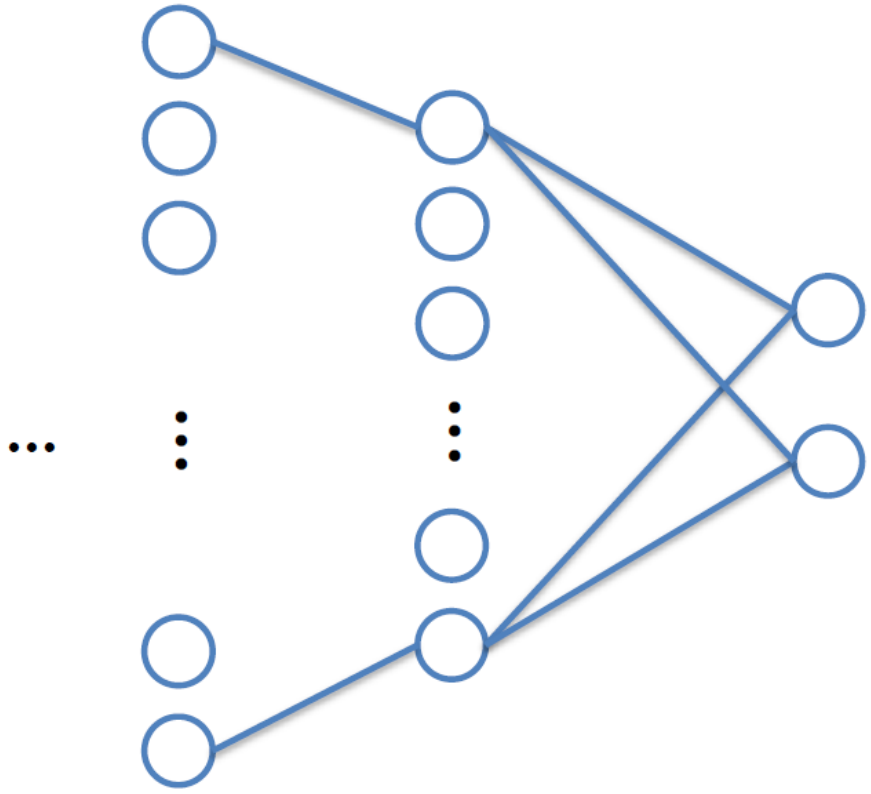
$\mathbf{x}^\top \mathbf{w}_k$	$e^{\mathbf{x}^\top \mathbf{w}_k}$	$\frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$	one-hot vector
강아지 0.37	1.4477	0.2296	0
고양이 1.58	4.8549	0.7704	1

수치 → exp → 확률 ↔ 확률

softmax!

Loss Function of 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

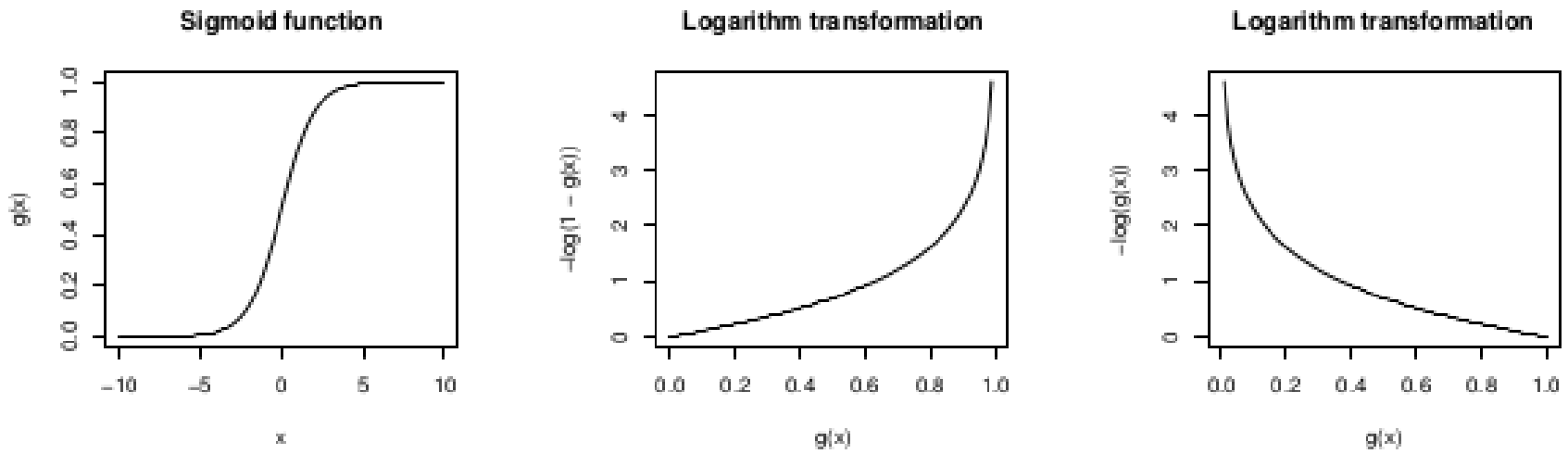
수치 → exp → 확률 ↔ 확률

softmax!

Loss Function of Logistic Regression

- Cross Entropy Loss!

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



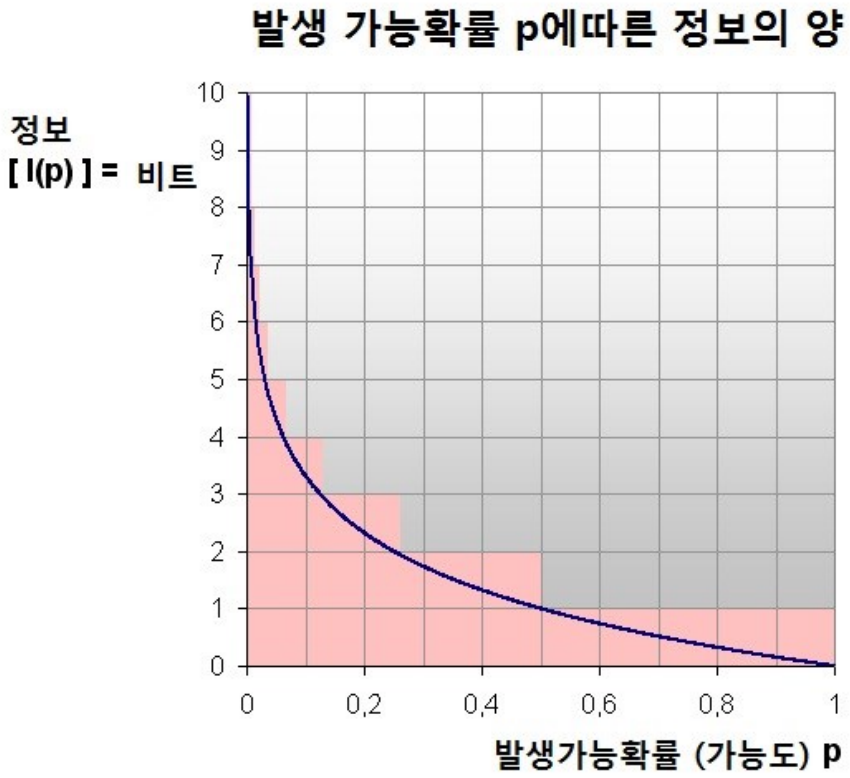
(a) *Sigmoid function.* **(b)** *Cost for* $y = 0$. **(c)** *Cost for* $y = 1$.

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

entropy

- Information Theory에서 Entropy란 정보량의 기댓값(평균)이다

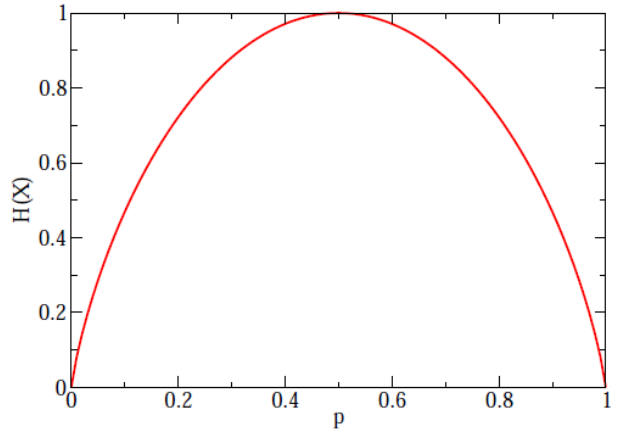
$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\ln(\mathrm{P}(X))].$$



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

Entropy

$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\ln(\mathrm{P}(X))].$$
$$= \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i)\log_b \mathrm{P}(x_i),$$



Ex) 동전던지기

P(앞) = ½
P(뒤) = ½

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

이 확률 분포를 표현하기 위하여 평균 1 bit가 필요하다는 의미!

Cross Entropy

True distribution p,
Model이 예측한 distribution을 q라고 하면

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

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

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

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

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

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

P 자체의 entropy(불변)

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

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 x (log0.4 + log0.4 + log0.1) = 1.38	1/3 x (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 x (log0.7 + log0.7 + log0.4) = 0.64	1/3 x (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

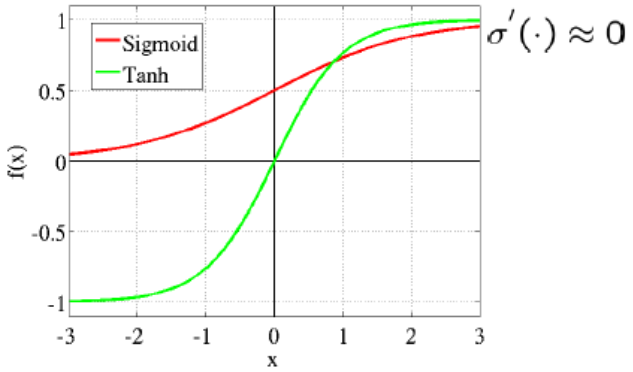
	Inference(softmax output)			True Label(onehot)			Correct?
	class1	class2	class3	class1	class2	class3	
model1	0.3	0.3	0.4	0	0	1	Yes
	0.3	0.4	0.3	0	1	0	Yes
	0.1	0.2	0.7	1	0	0	No
model2	0.1	0.2	0.7	0	0	1	Yes
	0.1	0.7	0.2	0	1	0	Yes
	0.3	0.4	0.3	1	0	0	No

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

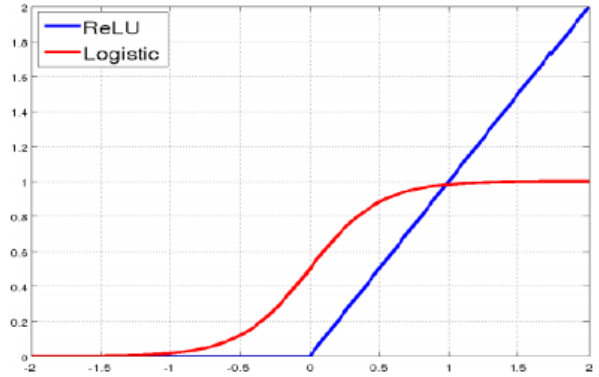
Multi-Layer, Multi-Class에 대한 학습법을 배웠으니 이제 Deep하게 쌓으면 다 잘 될....

Activation Function – ReLU

- ReLU를 activation function으로 사용 → sparse activation
- ReLU는 미분값이 0 아니면 1 → vanishing gradient 해결

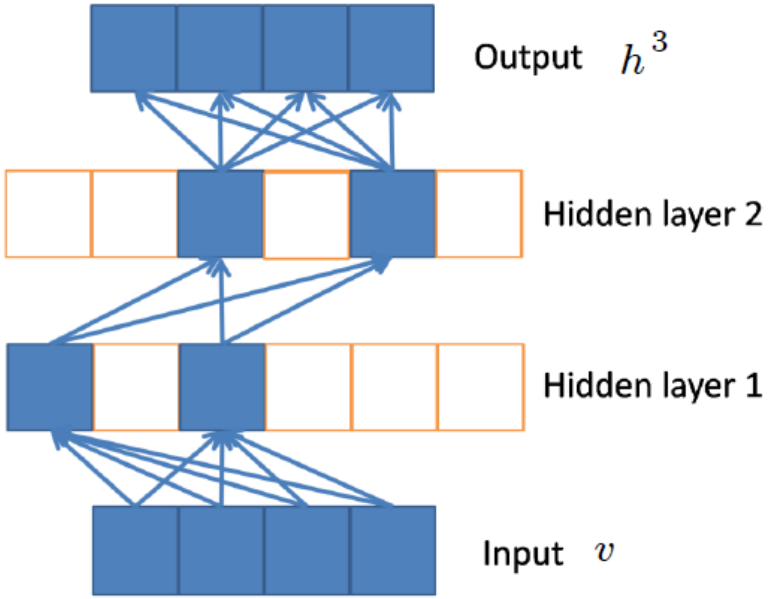


$\sigma'(\cdot) \approx 0$



$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

slope: 1

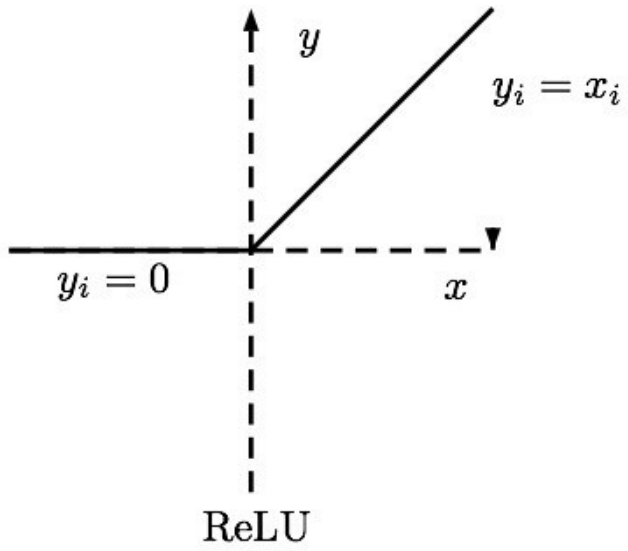


ReLU는 만능?

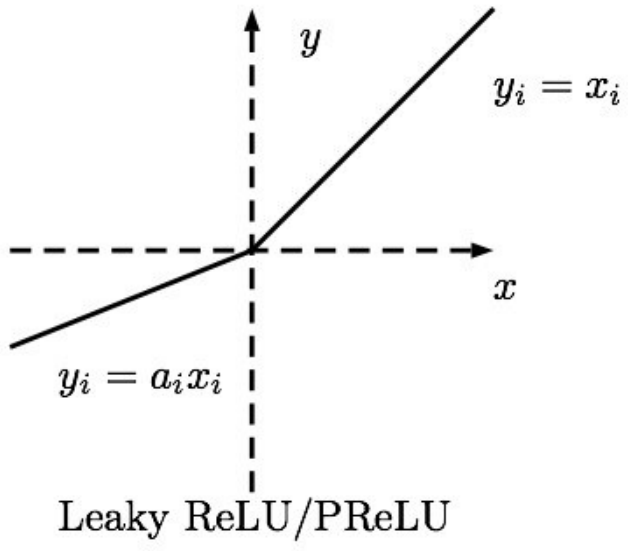
- Dying ReLU(Dying Neuron)
  - ReLU는 0 이하의 값을 무조건 버리기 때문에 gradient도 0이 되고 음수값에 대해서는 back propagation이 일어나지 않음
  - input에 따라서 output이 대부분(혹은 전부) 음수가 되는 경우에는 학습도 거의 일어나지 않음
  - Ex) $y = -x - 2$ 와 같이 $w = -1$, $b = -2$인 경우, -1~1 사이로 normalize된 input이 들어올 경우 output은 무조건 0 이하가 되어 영원히 weight가 update되지 않음

Friends of ReLU

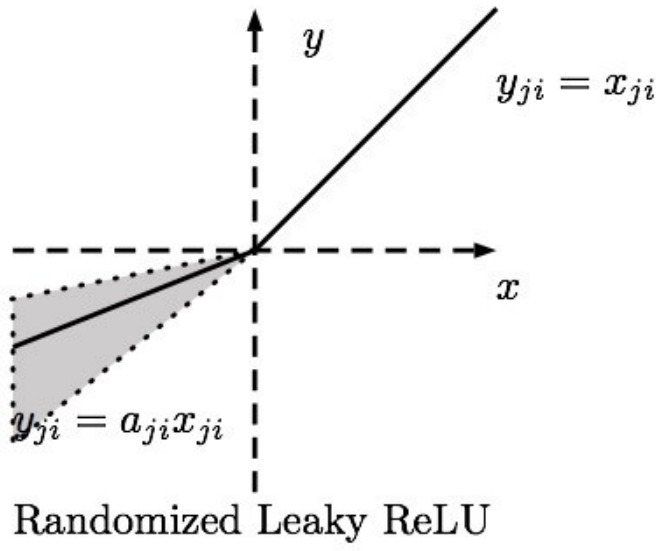
- X < 0 일 때도 weight가 update되도록 개선



ReLU

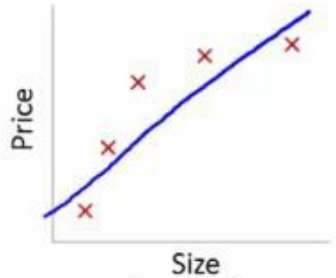


Leaky ReLU/PReLU



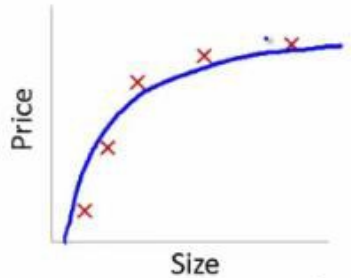
Randomized Leaky ReLU

Underfitting vs Overfitting



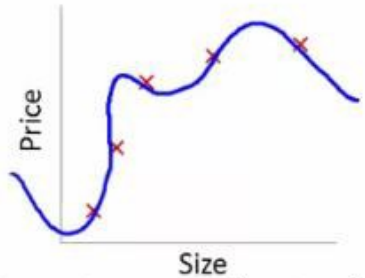
$\theta_0 + \theta_1 x$

High bias
(underfit)



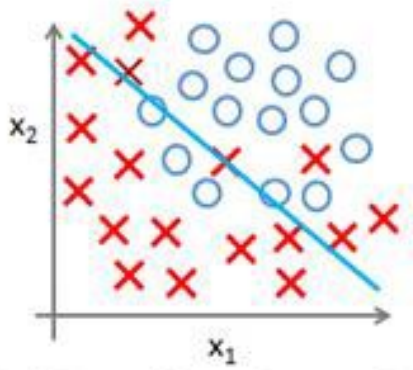
$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



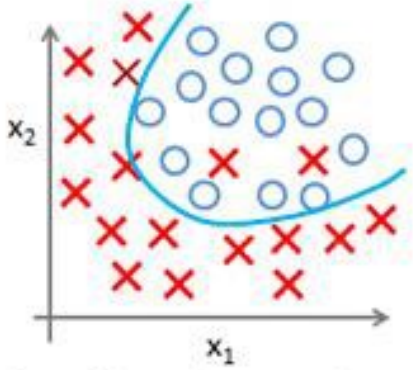
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

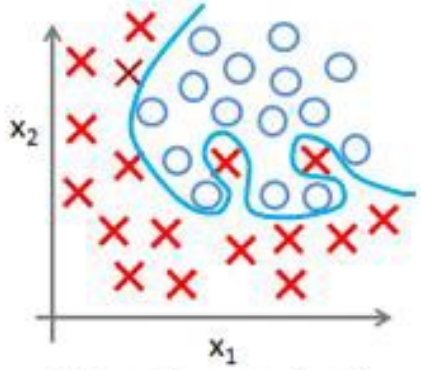


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

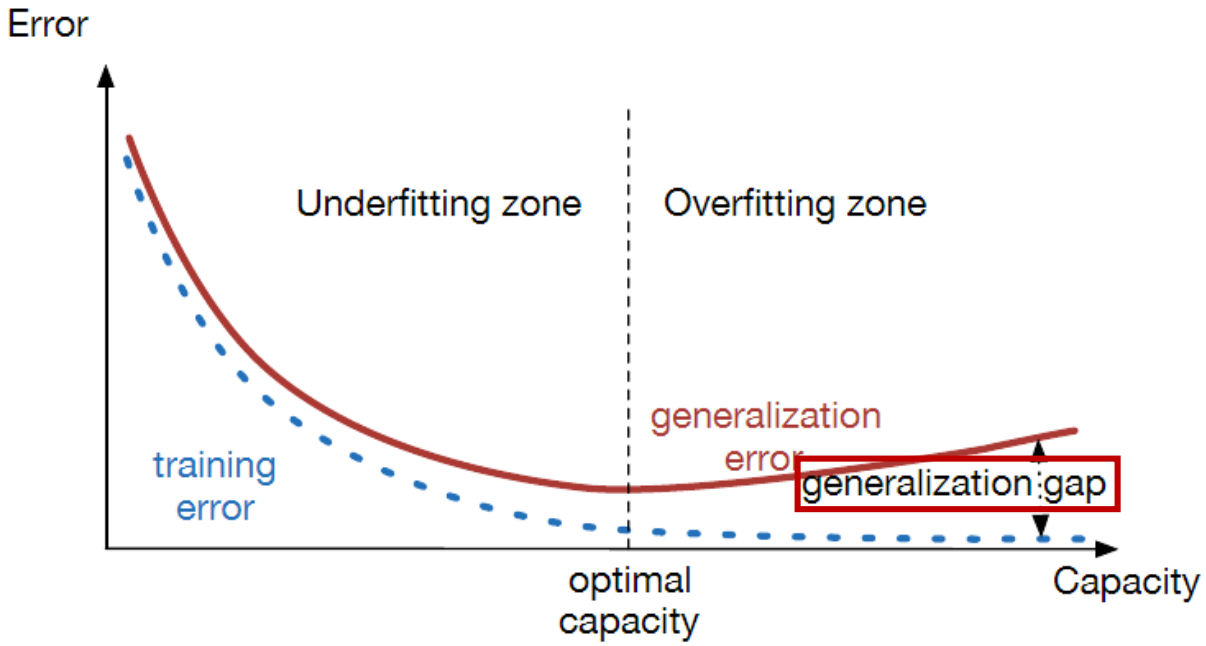


$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

Underfitting vs Overfitting



- test error

= train error + generalization gap

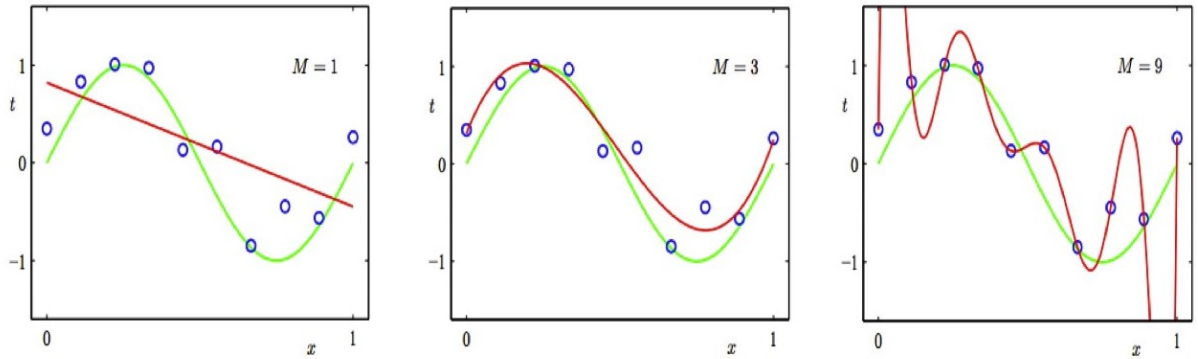
- Model capacity가 작으면 optimal capacity에 도달 자체가 불가능
- Model capacity가 너무 크면 overfitting이 일어남 (generalization gap이 커짐)

- Capacity는 크게하고 generalization gap을 줄이는 방법을 찾아보자 → deep learning!

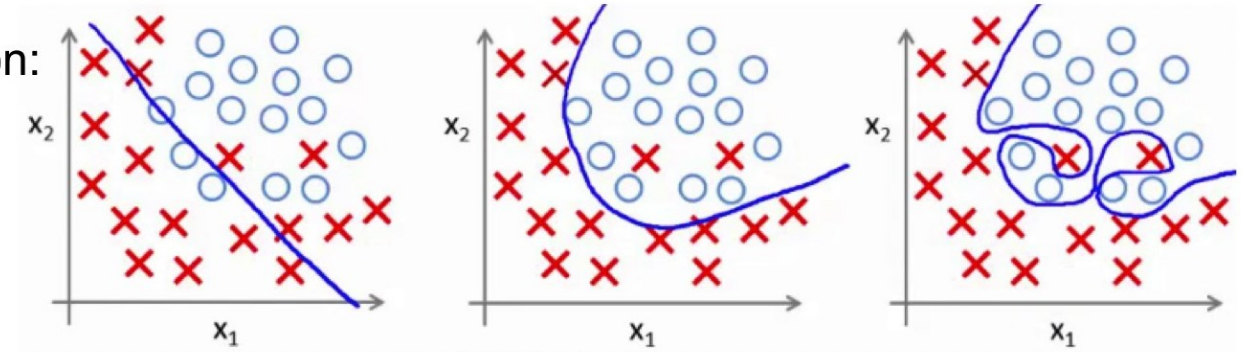
Fight Against Overfitting

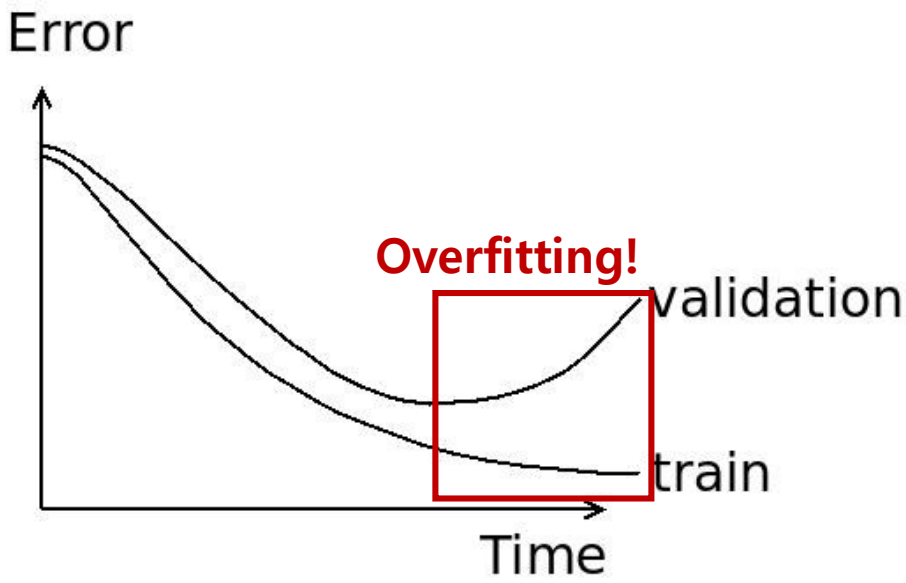
- Data가 많지 않아서 발생
- 학습한 data에만 최적화되어서, 학습하지 않은 data(test data)에 대한 추론 성능이 악화되는 현상

Regression:



Classification:





Regularization Method

- Dropout
- Weight Decay(L2 Regularization)

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2$$

L2 regularization	$R(W) = \sum_k \sum_l W_{k,l}^2$
L1 regularization	$R(W) = \sum_k \sum_l \lvert W_{k,l} \rvert$
Elastic net (L1 + L2)	$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + \lvert W_{k,l} \rvert$

- Batch Normalization
  - Benefits of BN
    - Increase learning rate
    - Remove dropout
    - Reduce L2 weight decay
    - Remove LRN

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

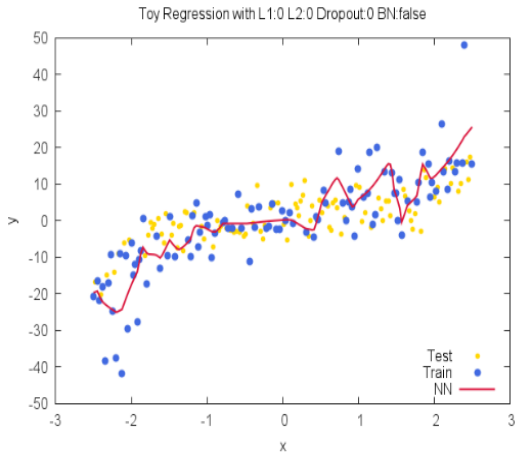
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

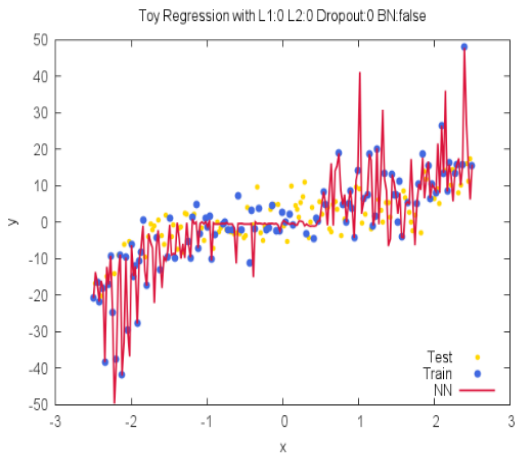
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

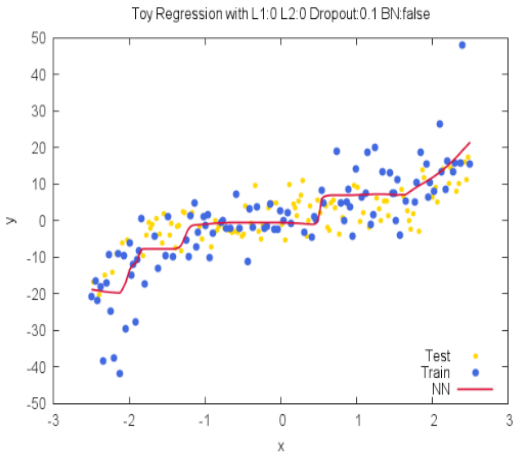
Regularization Methods

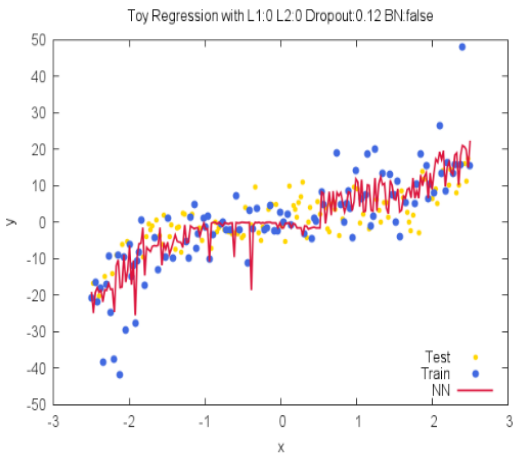
No Regularization



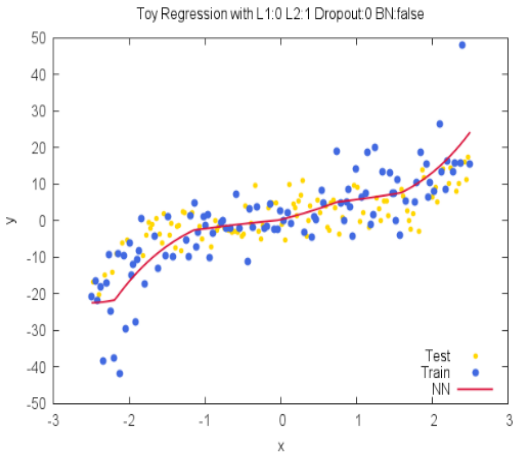


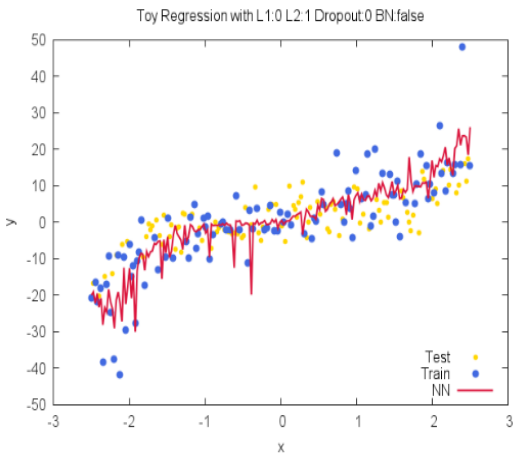
Dropout



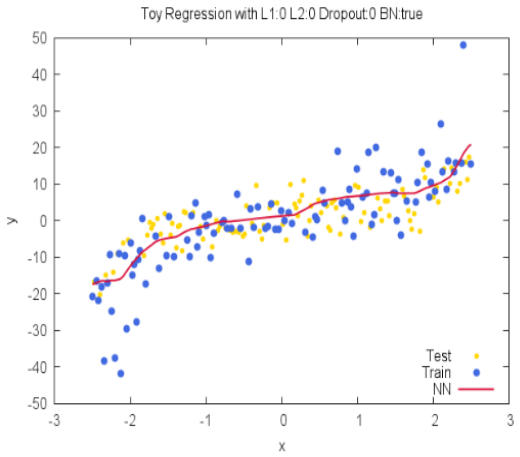


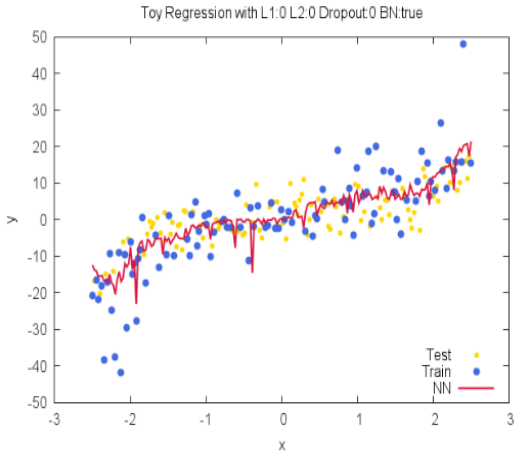
L2 Regularization



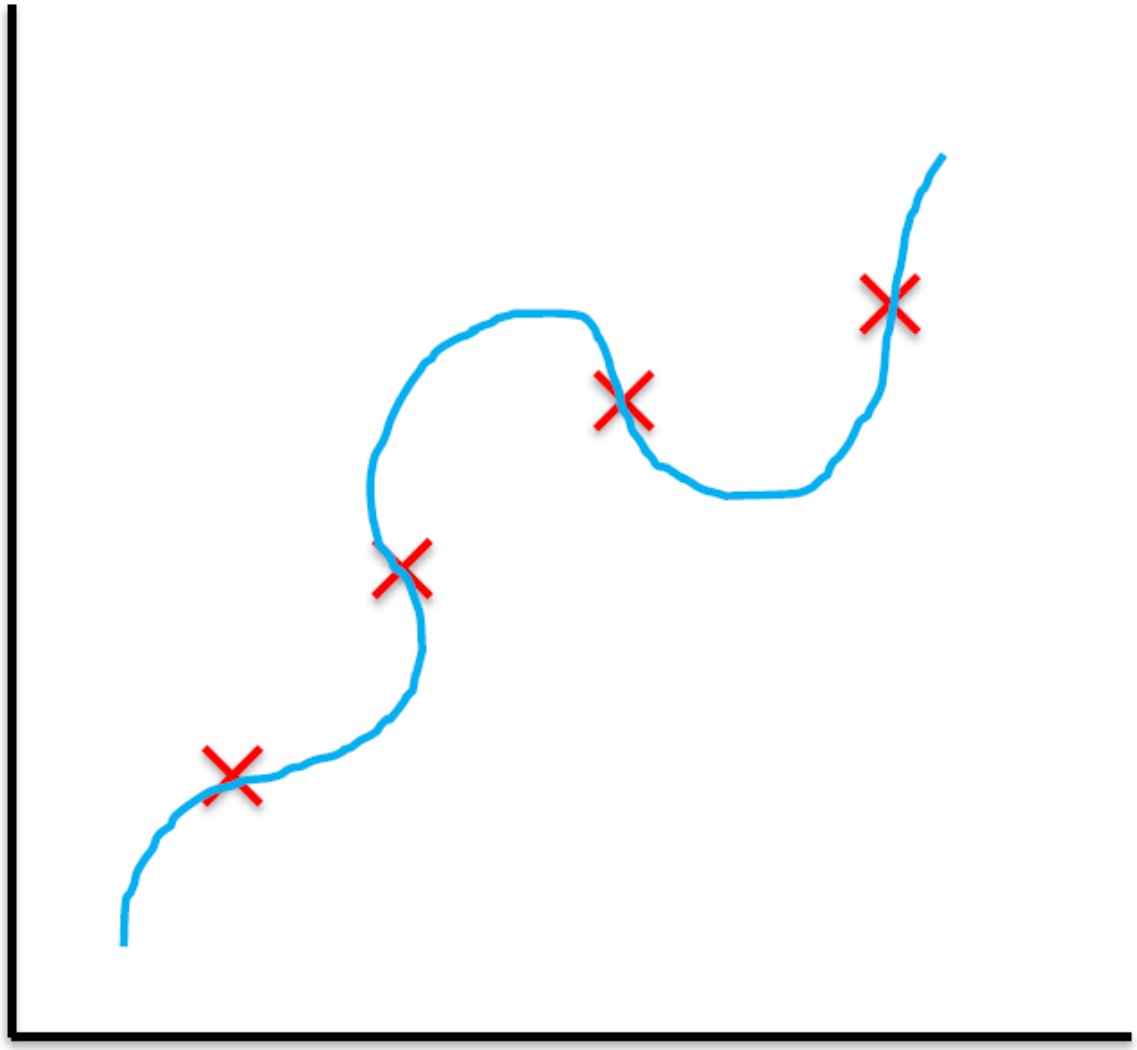


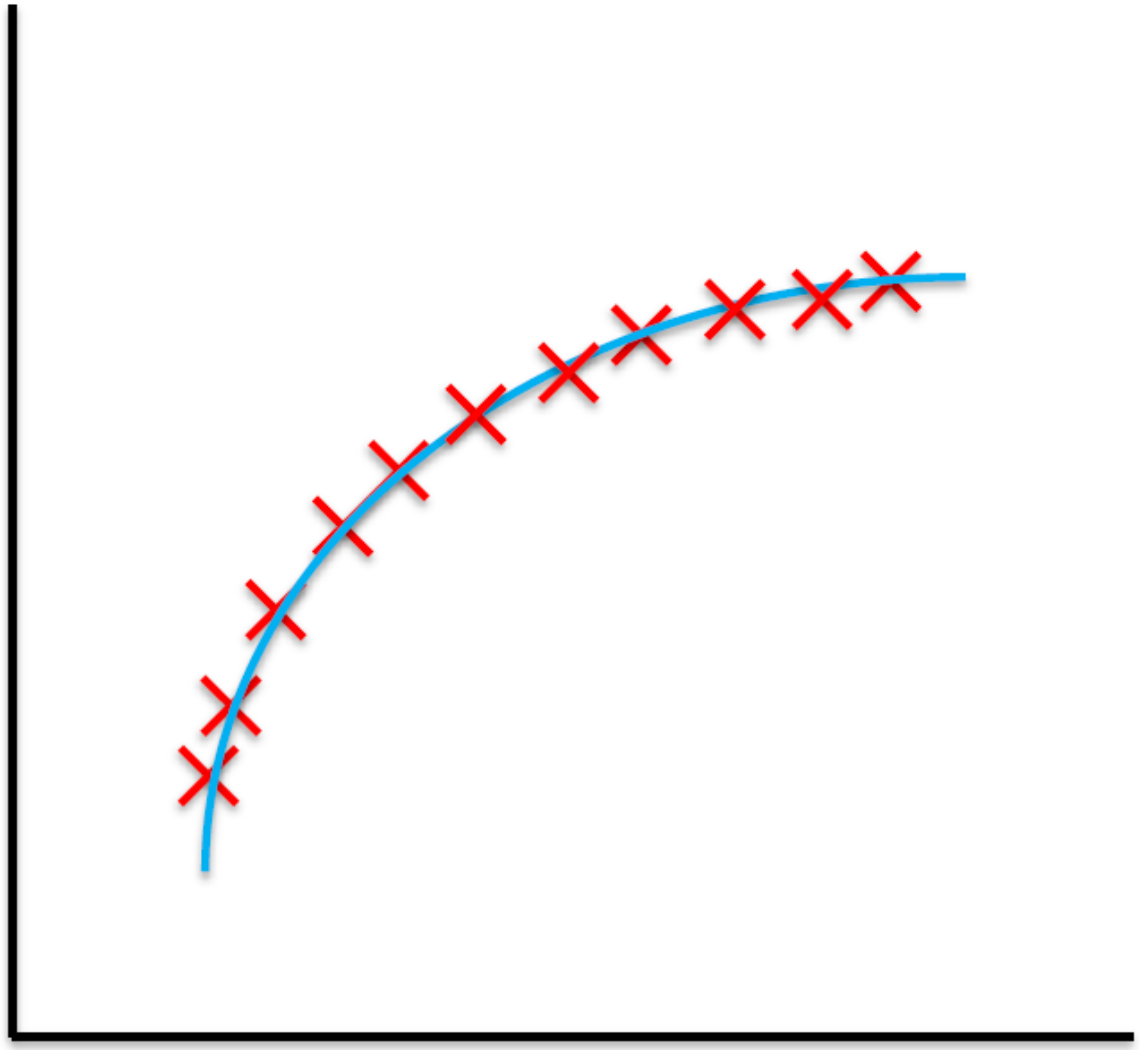
Batch Normalization





Data를 늘리면?





Data Augmentation

원본

Flip(LR)

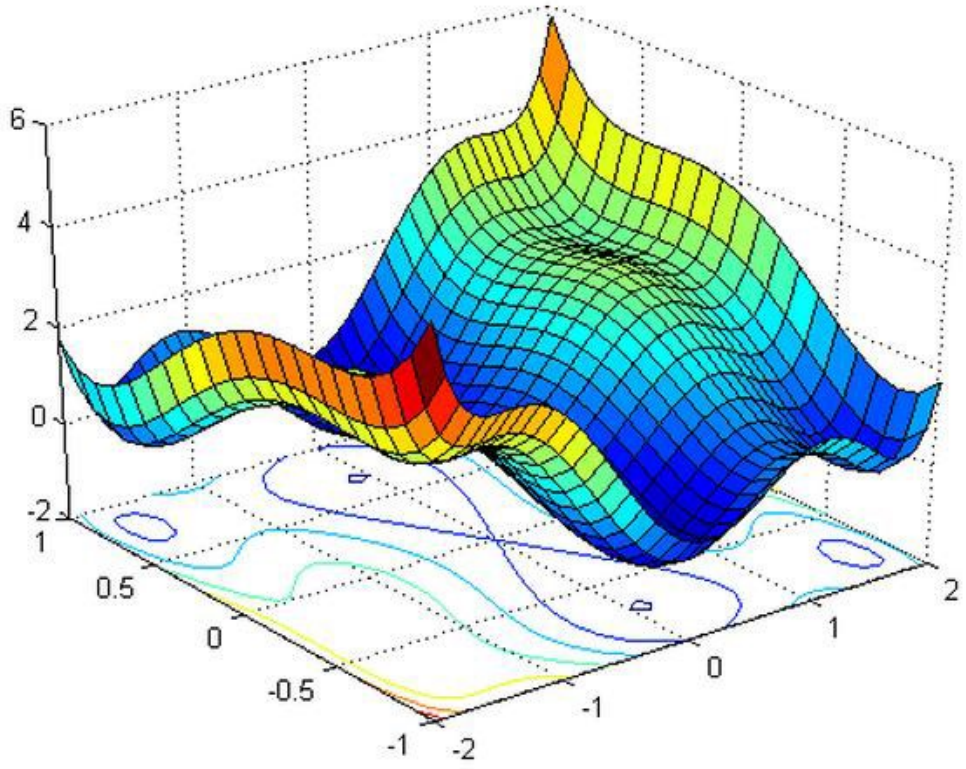
Flip(UD)

Translation

Rotate

CROP

Toward Global Minimum



- Global Minimum에 어떻게 도달할 수 있을까?
- 학습법은 Gradient Descent를 base로 하기 때문에 시작 지점이 중요하다
- 그런데 Global minimum의 위치를 모르기 때문에 좋은 시작점을 알 수 없음
- 학습할 때 input과 output의 variance를 비슷하게 맞춰보자!

→ Xavier Initialization / He Initialization

Weight Initialization

- Xavier Initialization
  - Activation function은 linear라고 가정하고, in/out의 variance를 같게 해보자

forward: $\mathrm{Var}(W_i) = \frac{1}{n} = \frac{1}{n_{\text{in}}}$

backward: $\mathrm{Var}(W_i) = \frac{1}{n_{\text{out}}}$

Xavier Initialization: $\mathrm{Var}(W_i) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$

Weight Initialization

- He Initialization
  - Activation function을 ReLU나 PReLU로 하고, variance를 같게 해보자

ReLU

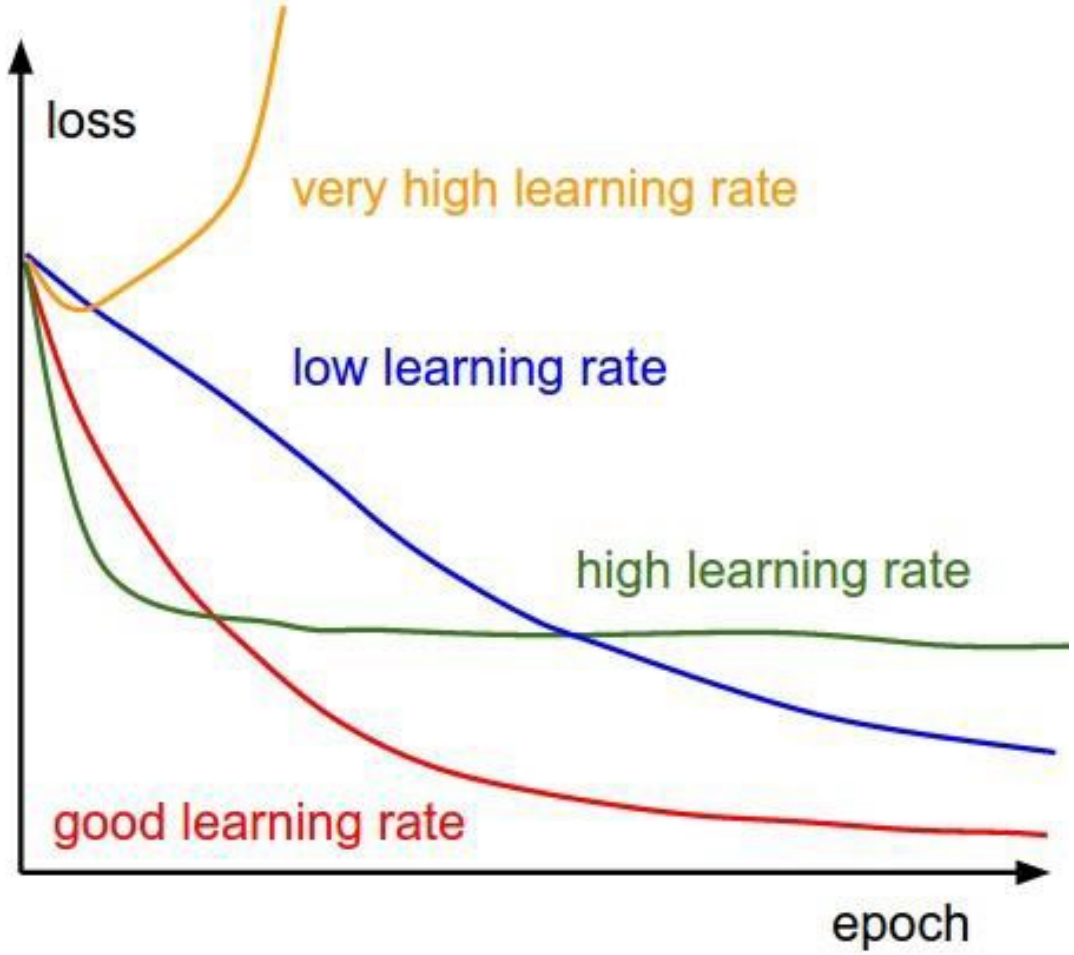
$$\begin{cases} Var[w_l] = \frac{2}{n_l} \implies \text{standard deviation (std)} = \sqrt{\frac{2}{n_l}} \\ W_l \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_l}}\right) \text{ and } \mathbf{b} = 0 \end{cases}$$

PReLU

$$\frac{1}{2}(1 + a^2) n_l Var[w_l] = 1$$

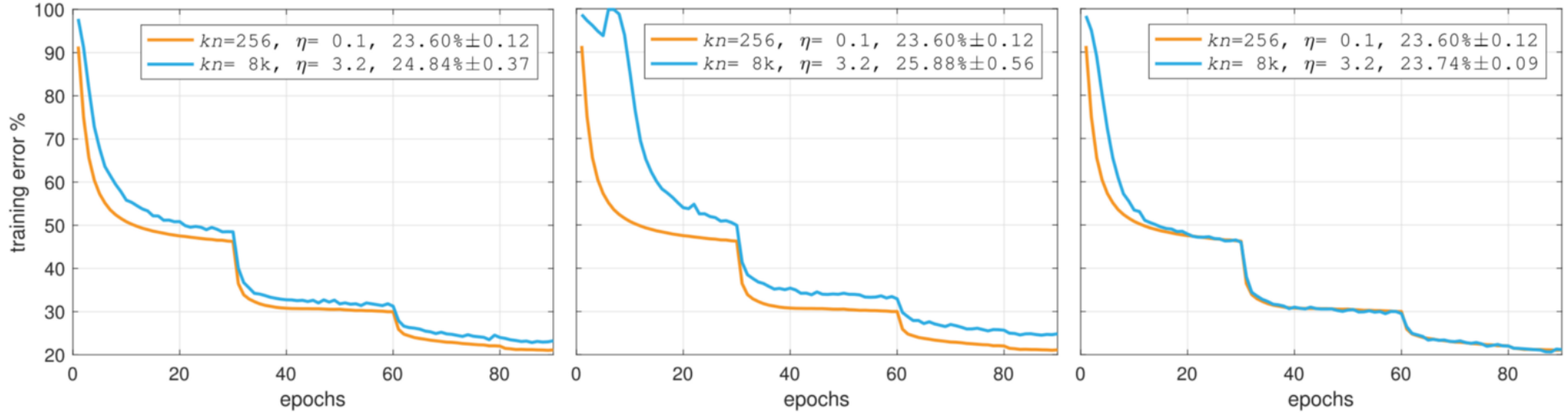
Learning Rate

- Initialization을 잘 해도 learning rate에 따라서 결과가 달라짐



Learning Rate Decay

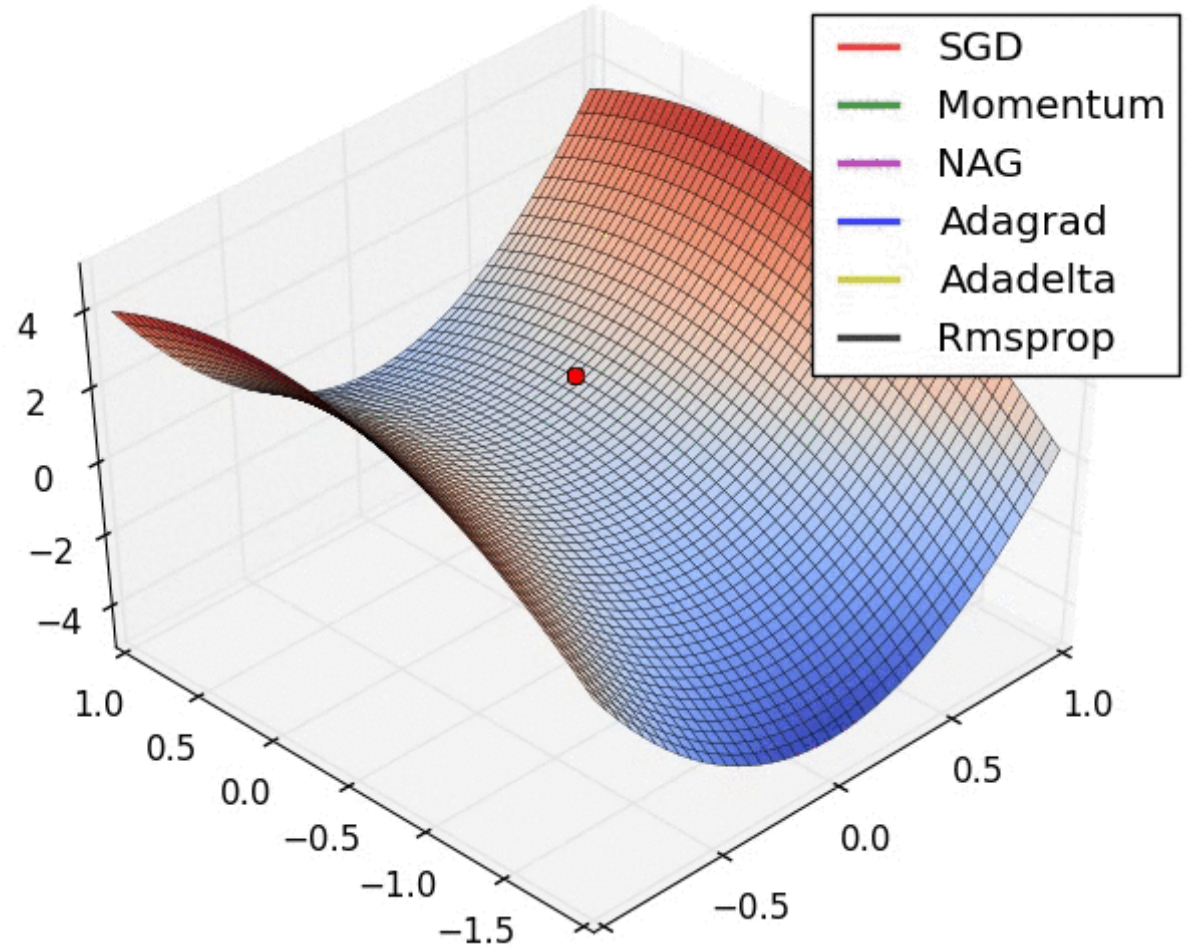
- Learning rate이 너무 크면 수렴을 못할 가능성이 있고, 너무 작으면 local minima 혹은 saddle point에서 못빠져나옴
- Learning Rate Decay
  - 처음에는 크게 움직이다가 일정 조건이 되면 learning rate을 낮춰서 점점 작게 움직이는 방법



(a) no warmup (b) constant warmup (c) gradual warmup

Optimization Methods

- Gradient descent로는 부족하다!



Optimization Methods

- We need more study!