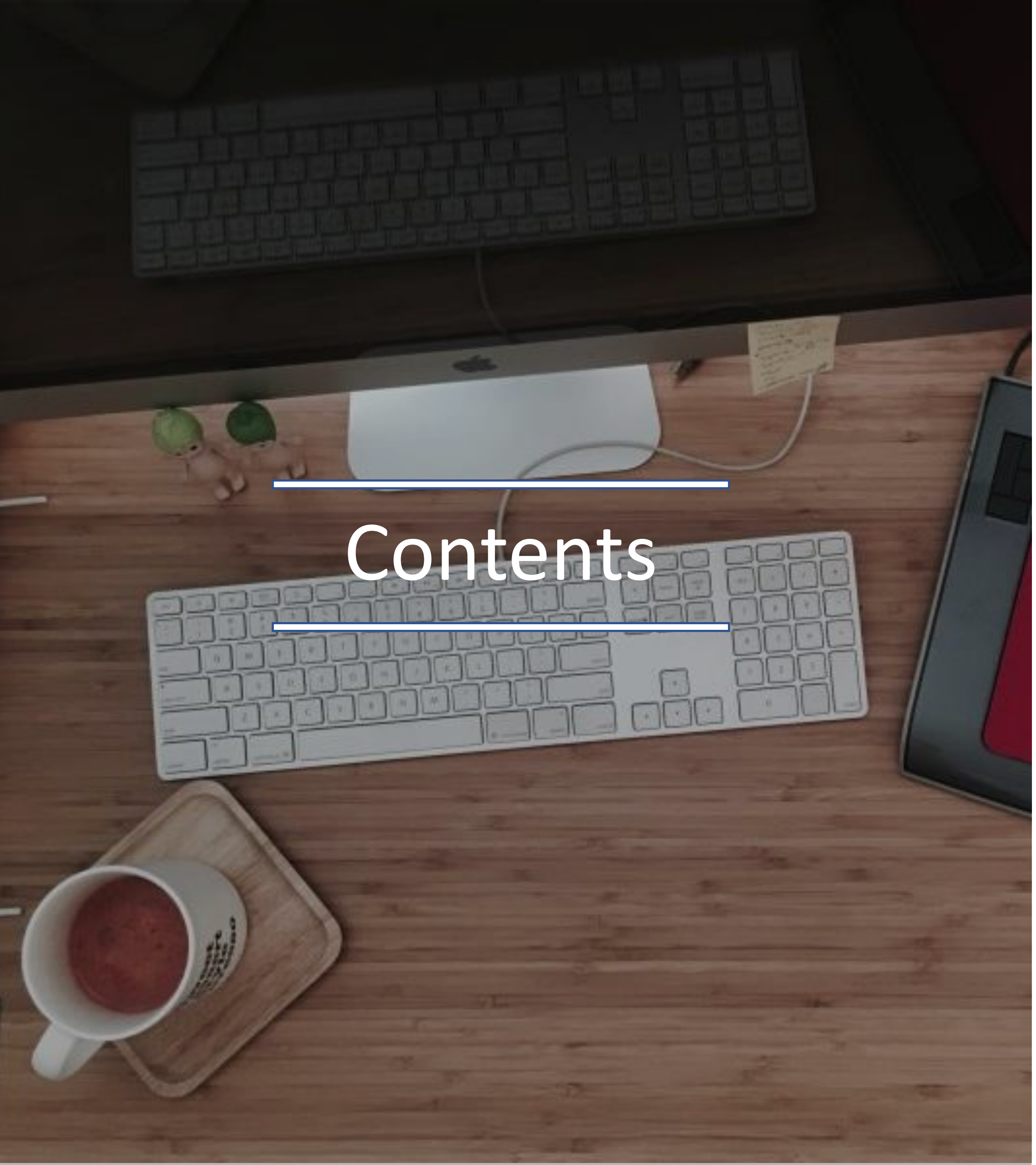
Apply to Machine Learning

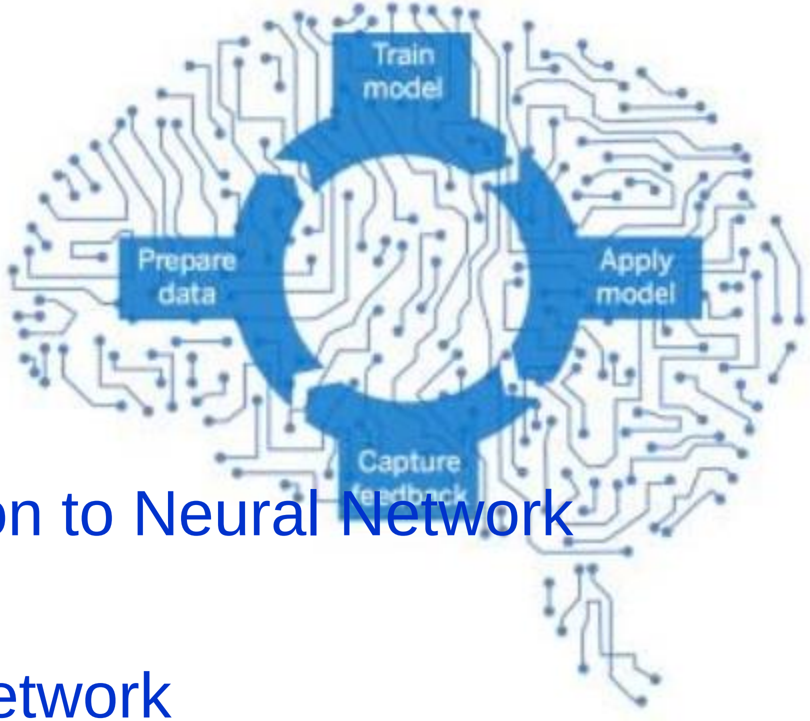
Week 4
Machine Learning

이선우(Seon Woo Lee)

인하대학교 HCI Lab
Inha University
Human-Computer Interaction Lab

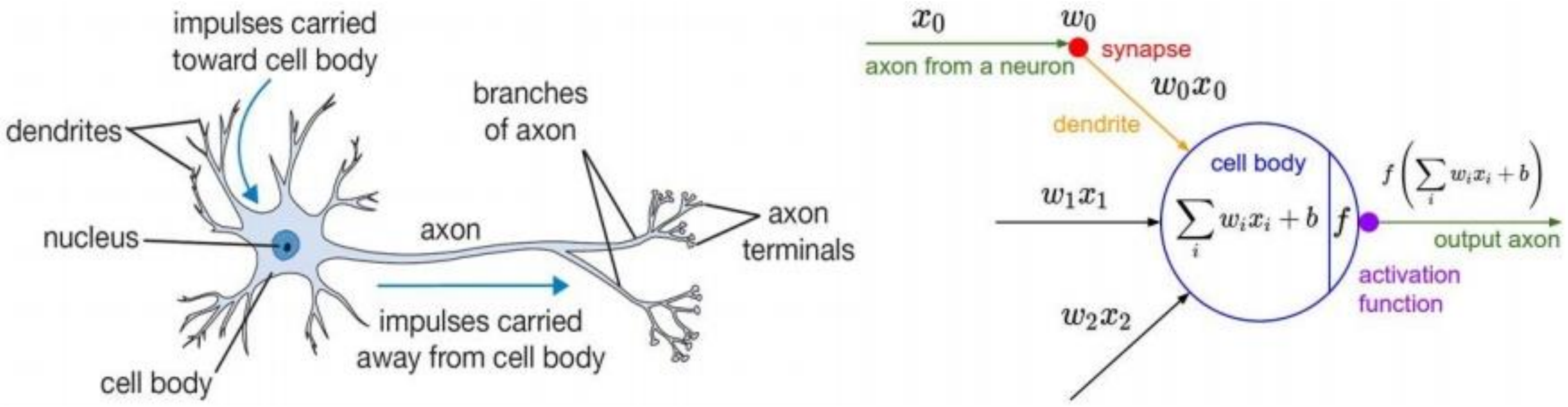


Contents



1 Propagation to Neural Network

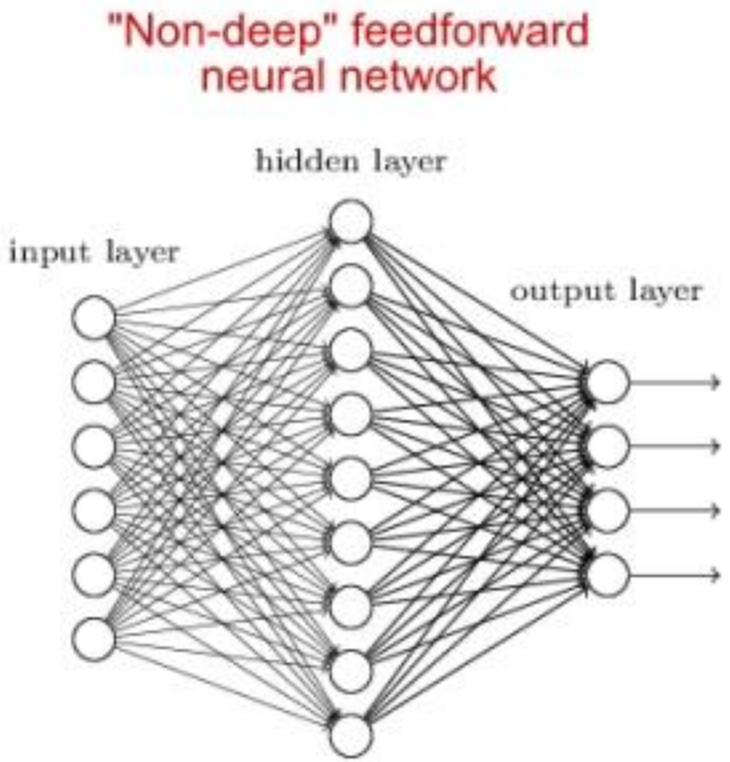
Neural Network

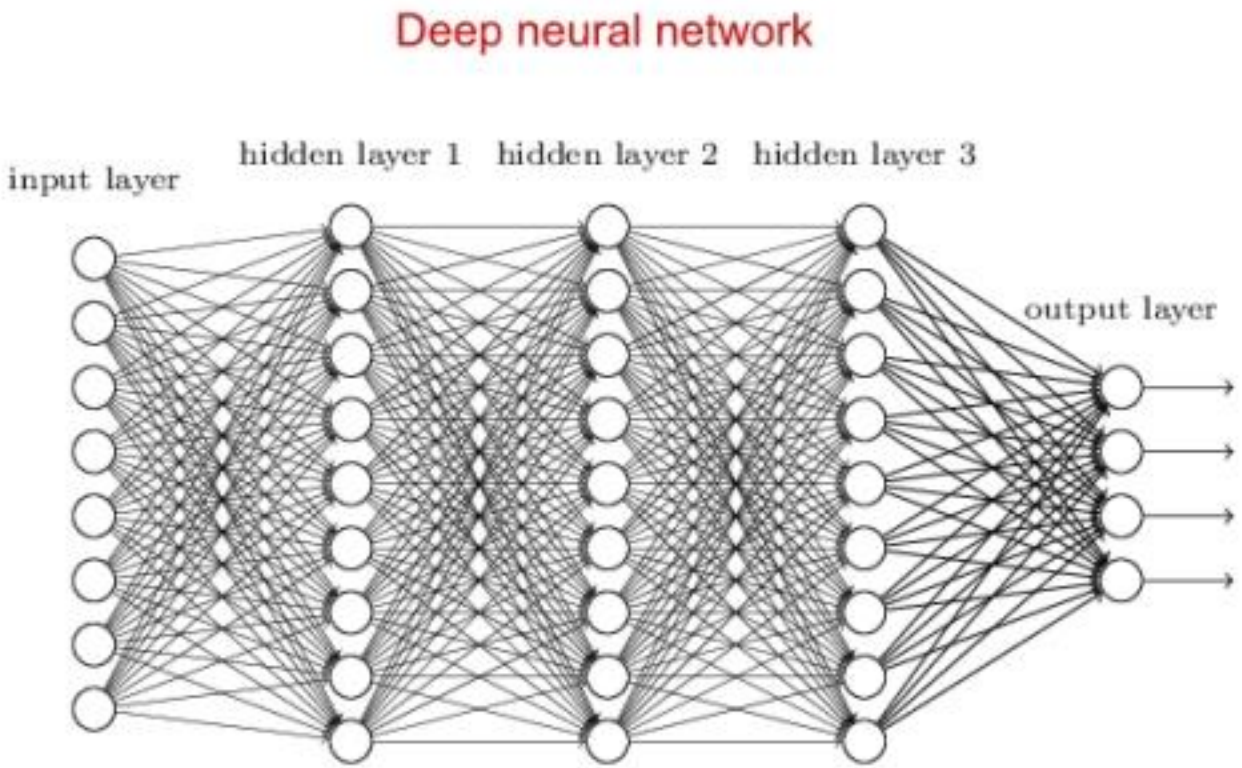


여러 자극이 들어오고 일정 기준을 넘으면 이를 다른 뉴런에 전달하는 구조

(출처 : http://cs231n.github.io/neural-networks-1/)

Neural Network





$$y = w2\left(act\left(w1 * input + b1\right)\right) + b2$$

$$y = w4(act\left(w3\left(act\left(w2\left(act\left(w1 * input + b1\right)\right) + b2\right)\right) + b3\right)\right) + b4$$

(출처 : http://neuralnetworksanddeeplearning.com/chap5.html)

Neural Network

$$y = W \cdot x + b$$

$$= \underset{m \times n}{\overset{X}{\begin{bmatrix} x_{00} & x_{01} & \cdots & \cdots \\ x_{10} & x_{11} & & \\ \vdots & & \ddots & \\ \vdots & & & x_{mn} \end{bmatrix}}} \underset{n \times l}{\overset{W}{\begin{bmatrix} w_{00} & w_{01} & w_{02} & \cdots \\ w_{10} & w_{11} & & \\ w_{20} & & \ddots & \\ \vdots & & & w_{nl} \end{bmatrix}}} + \underset{m \times 1}{\overset{b}{\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix}}}$$

Neural Network

$$y = W \cdot x + b$$

$$= \underbrace{\overset{x}{\begin{bmatrix} \boxed{x_{00} \quad x_{01} \quad \cdots} \\ x_{10} \quad x_{11} \\ \vdots \qquad \ddots \\ \vdots \qquad\qquad x_{mn} \end{bmatrix}}}_{m \times n} \underbrace{\overset{W}{\begin{bmatrix} w_{00} & w_{01} & w_{02} & \cdots \\ w_{10} & w_{11} & & \\ w_{20} & & \ddots & \\ \vdots & & & w_{nl} \end{bmatrix}}}_{n \times l} + \underbrace{\overset{b}{\begin{bmatrix} \boxed{b_0} \\ b_1 \\ \vdots \\ b_m \end{bmatrix}}}_{m \times 1}$$

Neural Network

$$y = act(wx+b)$$

$$= \text{activation}\left(\underset{m \times 1}{\begin{bmatrix} \quad wx + b \quad \end{bmatrix}}\right)$$

Neural Network

만약 activation function 이 없다면 아래의 식은 결국 linear function.

$$y = w4(act\left(w3\left(act(w2(act(w1 * input + b1)) + b2)\right) + b3)\right) + b4$$

Neural Network

만약 activation function 이 없다면 아래의 식은 결국 linear function.

$$y = w4(act\left(w3\left(act\left(w2\left(act\left(w1 * input + b1\right)\right) + b2\right)\right) + b3\right)) + b4$$

activation function 으로 non-linearity 를 추가해야 함

Neural Network

만약 activation function 이 없다면 아래의 식은 결국 linear function.

$$y = w4(act\left(w3\left(act(w2(act(w1 * input + b1)) + b2)\right) + b3)\right) + b4$$

activation function 으로 non-linearity 를 추가해야 함

그렇다면 어떤 activation function 을 써야 할까 ?

Neural Network

- Step function

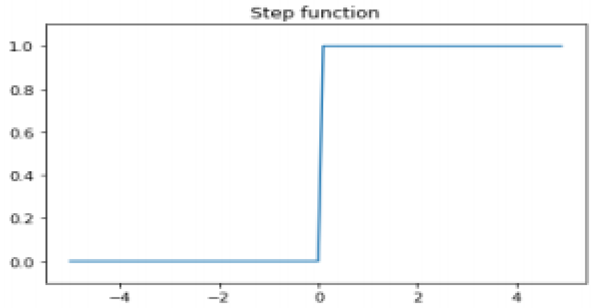
$$h(x) = \begin{cases} 0 \ (x \leq 0) \\ 1 \ (x > 0) \end{cases}$$

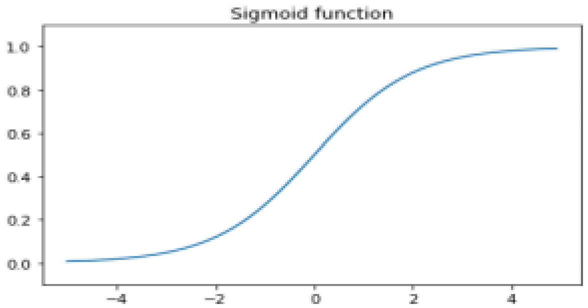
- Sigmoid

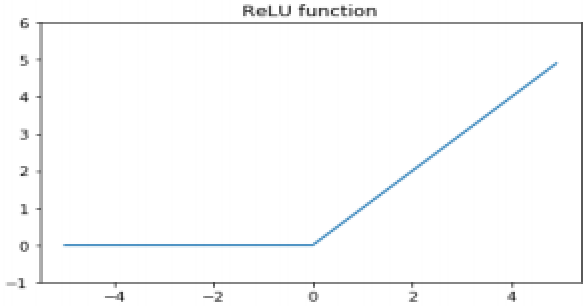
$$h(x) = \frac{1}{1 + \exp(-x)}$$

- ReLU (Rectified Linear Unit)

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







활성화 함수로는 반드시

비선형 함수를 사용한다

→ 선형 함수는 층을 깊게

하더라도 의미가 없기 때문

Neural Network

- Step function

```python
def step_function(x):
    return np.array(x > 0, dtype=np.int)
```

$x > 0$ 의 True/False를 $(numpy)int$로 변환하여 0 또는 1의 값으로 return

- Sigmoid

```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

$sigmoid(x) = \frac{1}{1+\exp(-x)}$ 식의 값을 return

- ReLU (Rectified Linear Unit)

```python
def relu(x):
    return np.maximum(0, x)
```

0과 x를 비교하여 큰 값을 출력한다.

$x >= 0$일 때는 x를, $x < 0$일 때는 0을 출력

Neural Network

- Softmax 함수

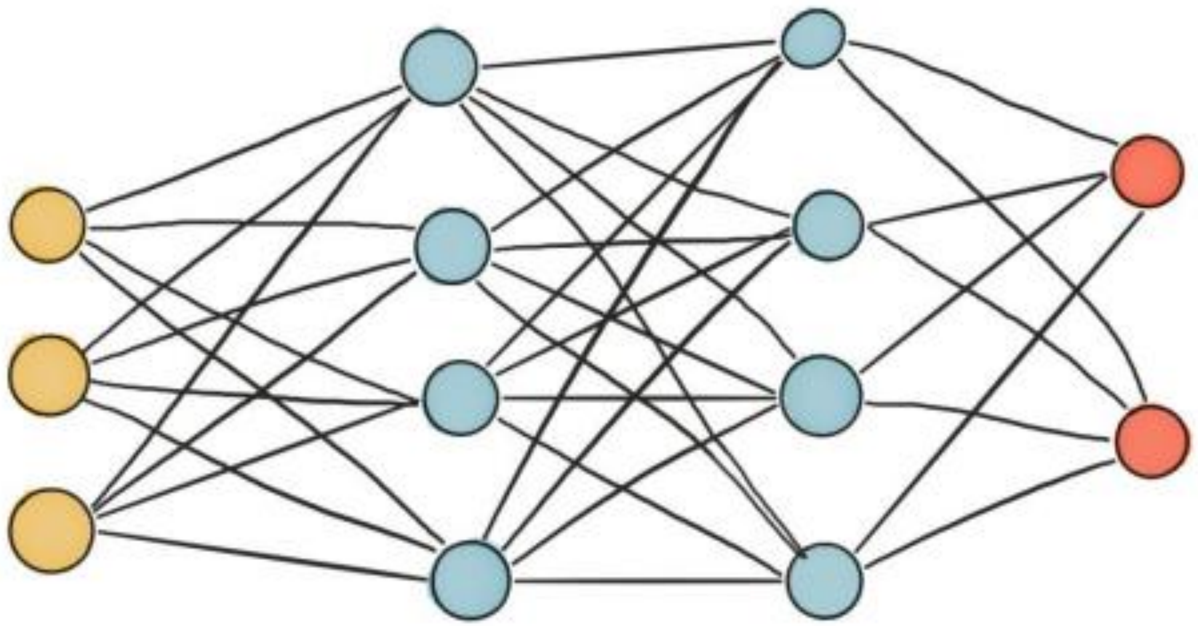
```python
def softmax(x):
    c = np.max(x)
    exp_x = np.exp(x - c) # overflow prevention
    sum_exp_x = np.sum(exp_x)
    return exp_x / sum_exp_x
```

- 지수 함수(exp)를 사용하기 때문에 오버플로 (inf 값 발생)가 날 수 있다.

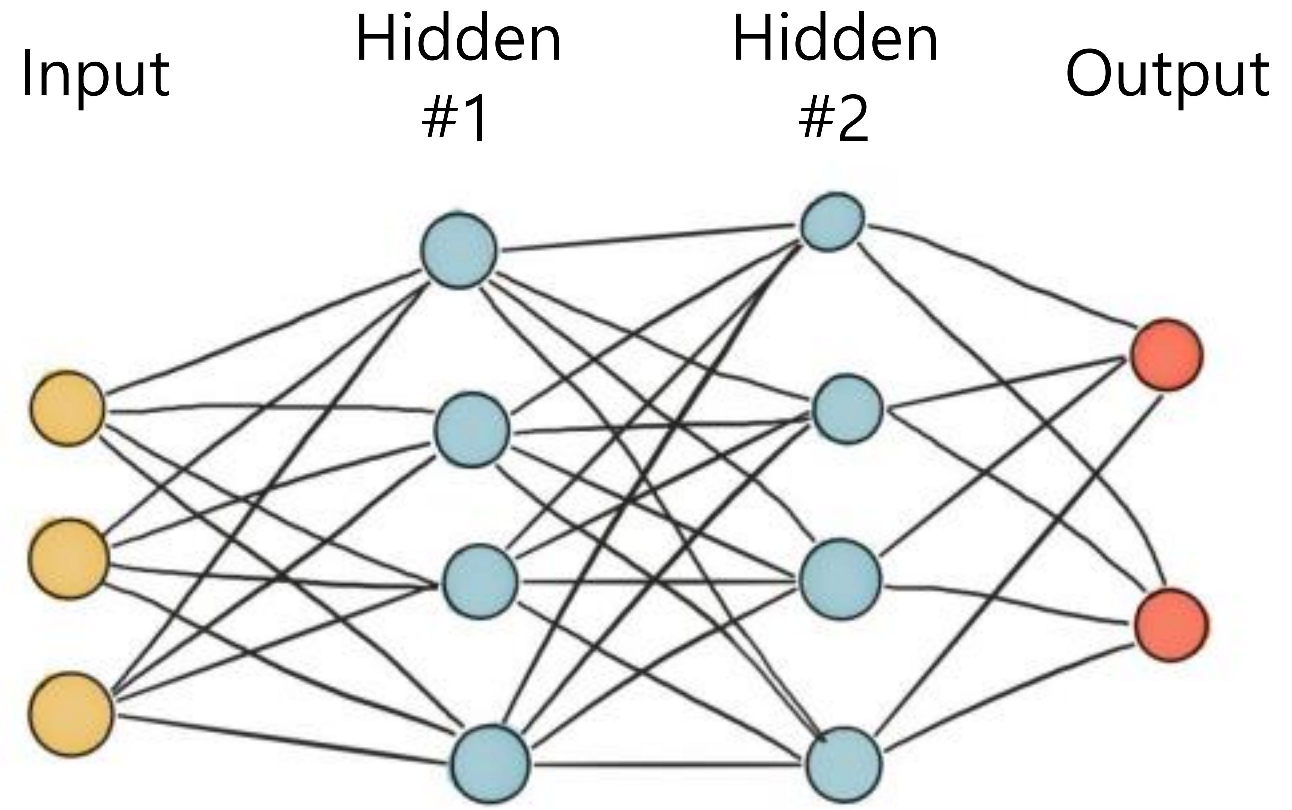
  => x중에서의 $np.max$값 c를 빼서 값이 무수히 커지는 것을 막아준다.

- $softmax(x) = \frac{\exp(x_k - C)}{\sum_{i=1}^{n} \exp(x_i - C)}$ 의 값을 return

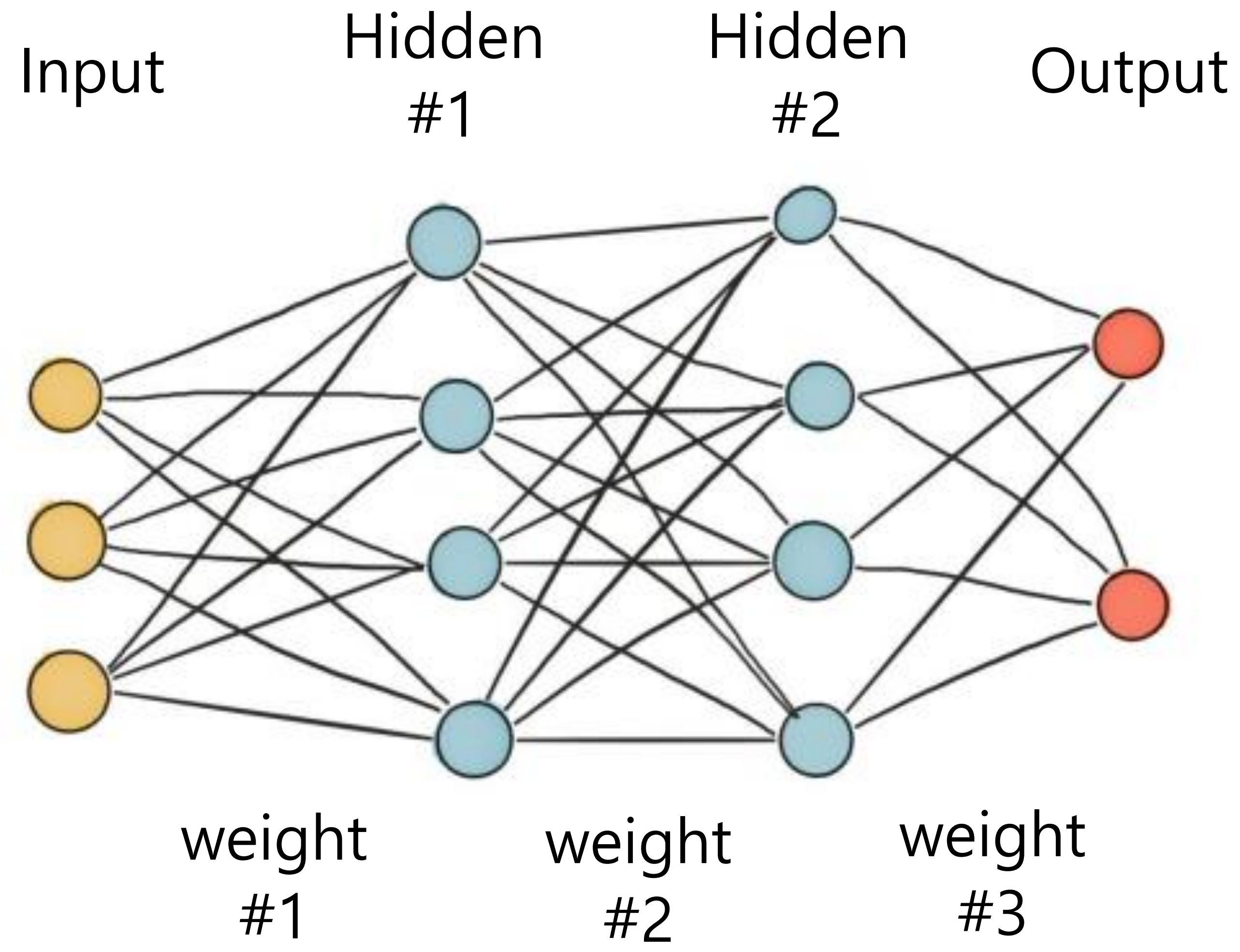
Forward & Back Prop.



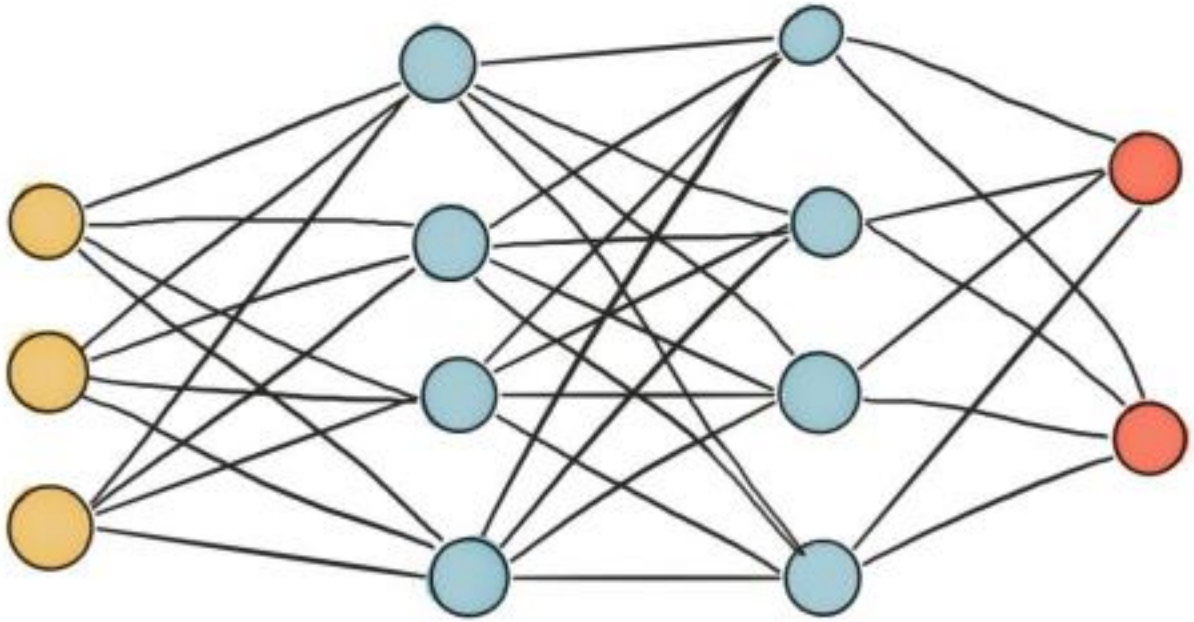
Forward & Back Prop.



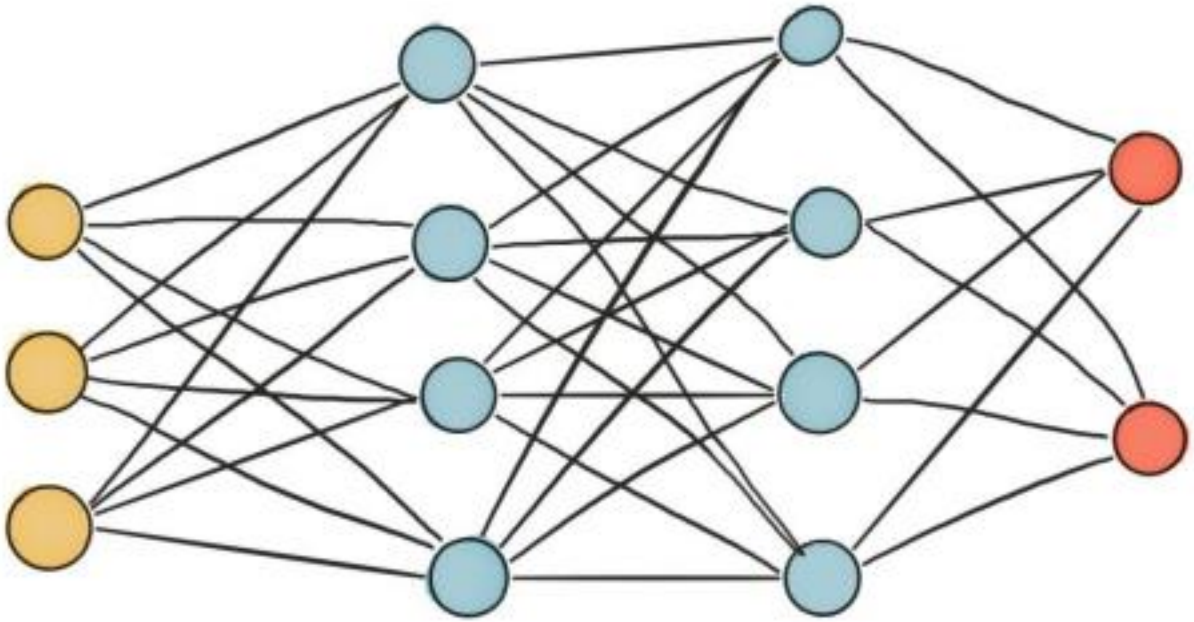
Forward & Back Prop.



Forward & Back Prop.

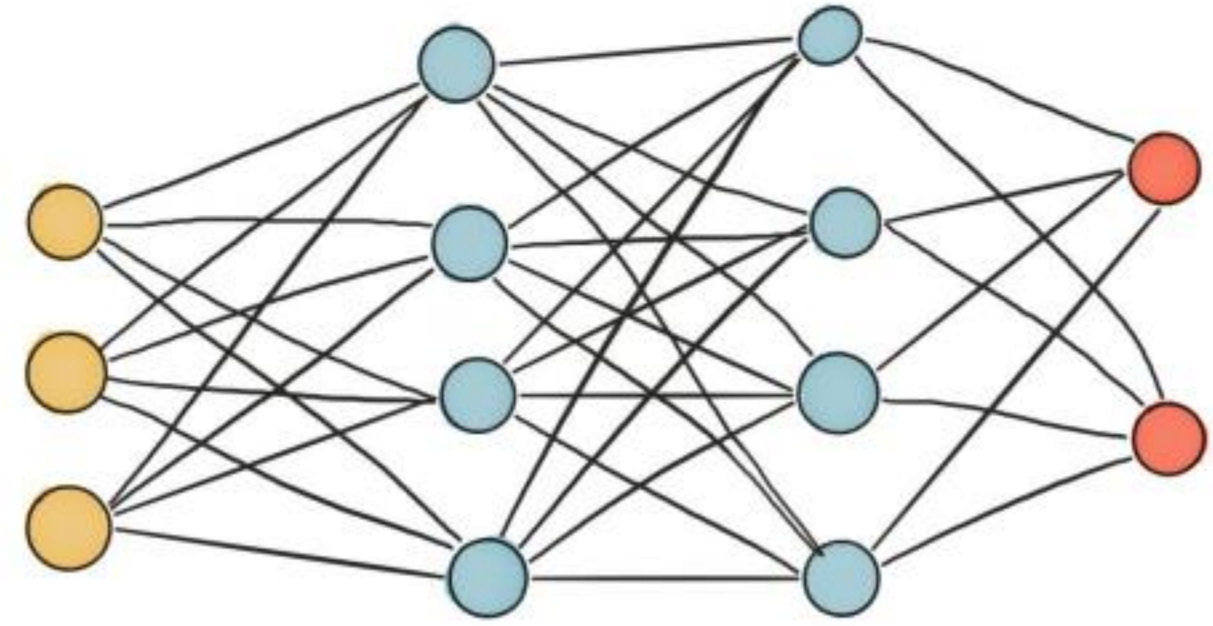


Forward & Back Prop.



$$\begin{bmatrix} w_{oo} & w_{01} & w_{02} & w_{03} \\ w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \end{bmatrix} \times \begin{bmatrix} w_{oo} & w_{01} & w_{02} & w_{03} \\ w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{bmatrix} \times \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \\ w_{30} & w_{31} \end{bmatrix}$$

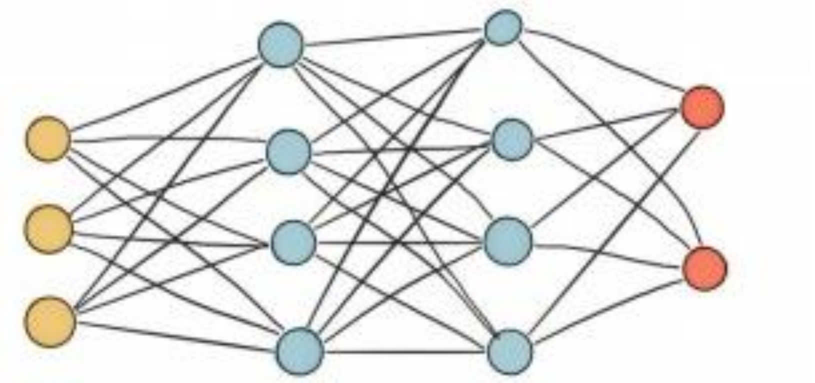
Forward & Back Prop.



$$\begin{bmatrix} w\,oo & w\,01 & w\,02 & w\,03 \\ w\,10 & w\,11 & w\,12 & w\,13 \\ w\,20 & w\,21 & w\,22 & w\,23 \end{bmatrix} \times \begin{bmatrix} w\,oo & w\,01 & w\,02 & w\,03 \\ w\,10 & w\,11 & w\,12 & w\,13 \\ w\,20 & w\,21 & w\,22 & w\,23 \\ w\,30 & w\,31 & w\,32 & w\,33 \end{bmatrix} \times \begin{bmatrix} w\,00 & w\,01 \\ w\,10 & w\,11 \\ w\,20 & w\,21 \\ w\,30 & w\,31 \end{bmatrix}$$

3x4 4x4 4x2

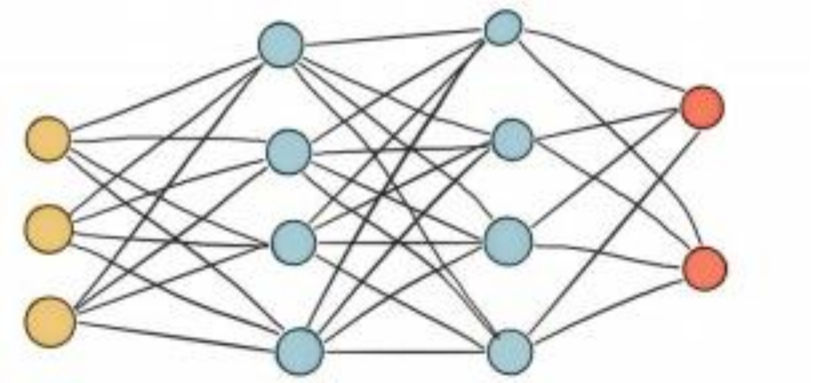
Forward & Back Prop.



$$y^* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

쉽게 이해되도록
loss = 예측값 - 실제로 설정

Forward & Back Prop.

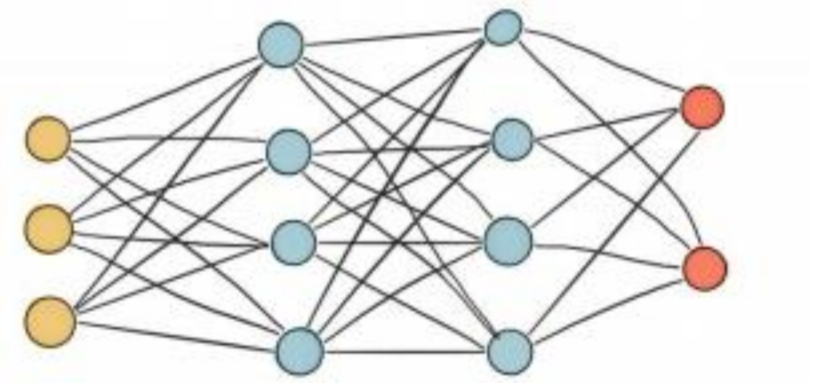


쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y^* = w3 * sig\ \ w2 * sig\ \ w1 * x + b1\ \ + b2\ \ + b3$$

$$\begin{aligned} loss\ \ &= y^* - y \\ &= w3 * sig\ \ w2 * sig\ \ w1 * x + b1\ \ + b2\ \ + b3 - y \end{aligned}$$

Forward & Back Prop.



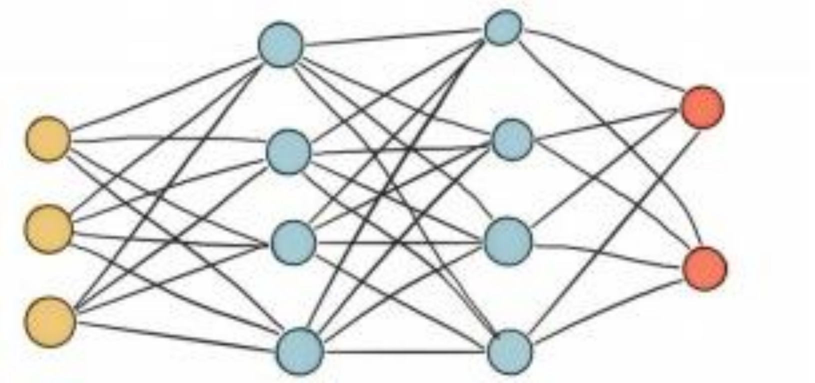
쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y^* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

$$\begin{aligned} loss &= y^* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

Forward & Back Prop.

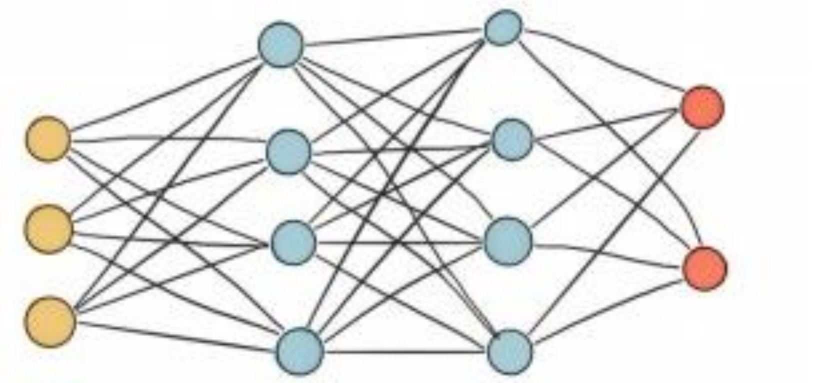


쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$\frac{\partial loss}{\partial w3} = sig\,(w2 * sig\,(w1 * x + b1)\ + b2)$$

$$\frac{\partial loss}{\partial b3} = 1$$

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

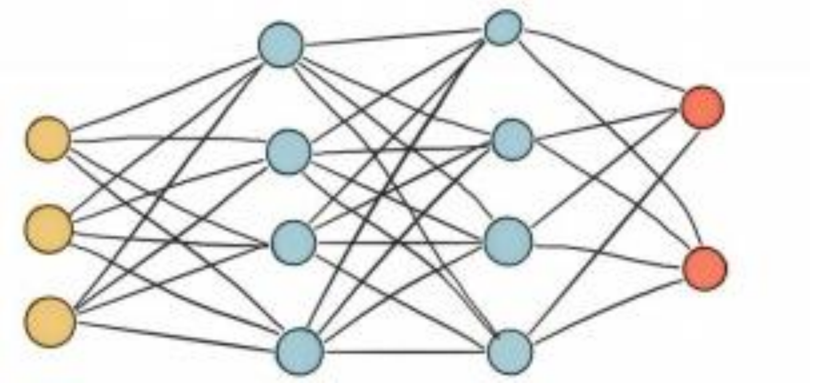
$$\begin{aligned} loss &= y* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

$$\frac{\partial loss}{\partial b3} = 1$$

$$\frac{\partial loss}{\partial w2} = ??$$

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

$$\begin{aligned} loss &= y* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

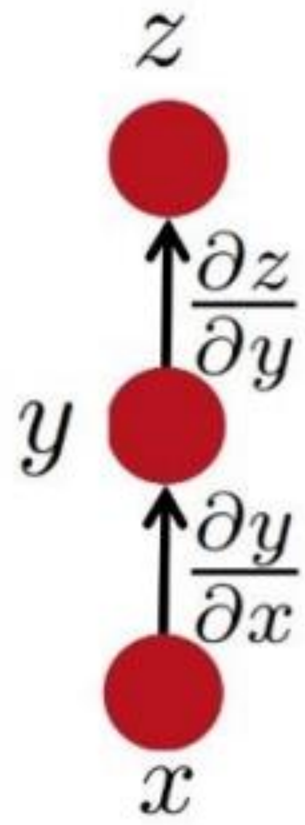
$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

$$\frac{\partial loss}{\partial b3} = 1$$

$$\frac{\partial loss}{\partial w2} = chain\ rule!!$$

Forward & Back Prop.

Simple Chain Rule



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

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

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

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

(출처 :https://www.slideshare.net/roelofp/041114-dl-nlpwordembeddings)

Forward & Back Prop.

Backpropagation: a simple example

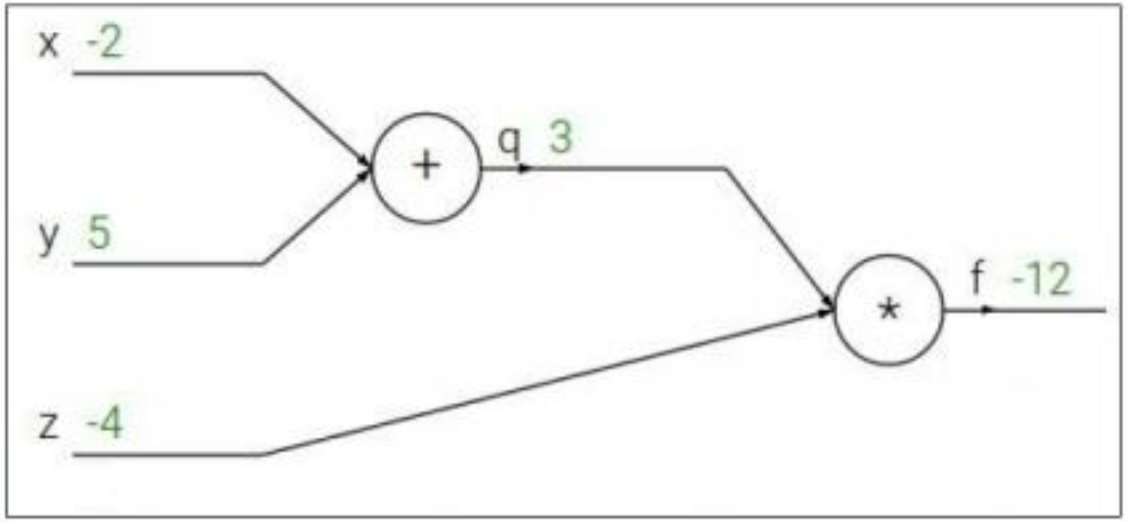
$$f(x, y, z) = (x + y)z$$

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



(출처 : cs231n_lecture4 p.13)

Forward & Back Prop.

Backpropagation: a simple example

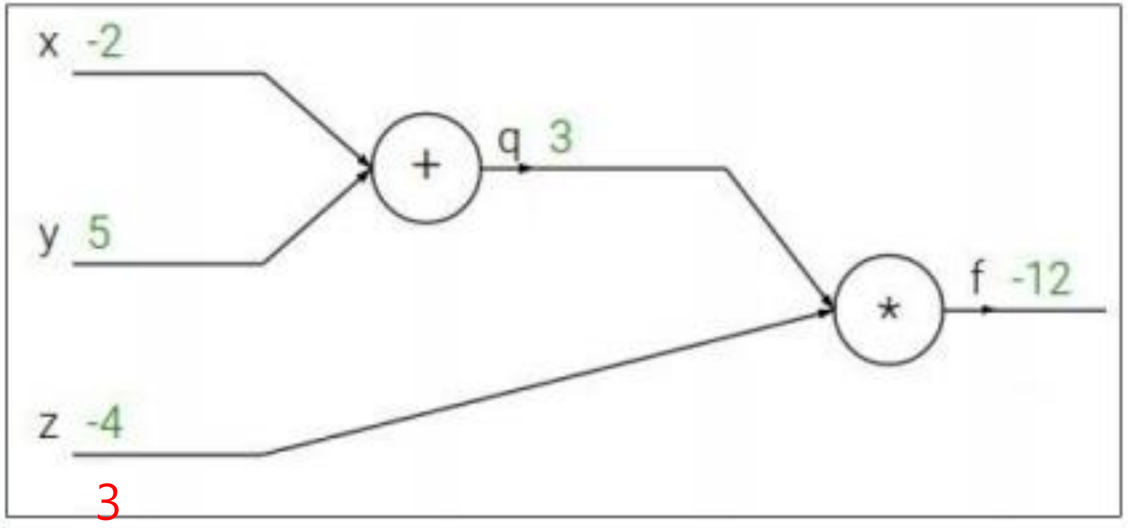
$$f(x, y, z) = (x + y)z$$

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



$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

Forward & Back Prop.

Backpropagation: a simple example

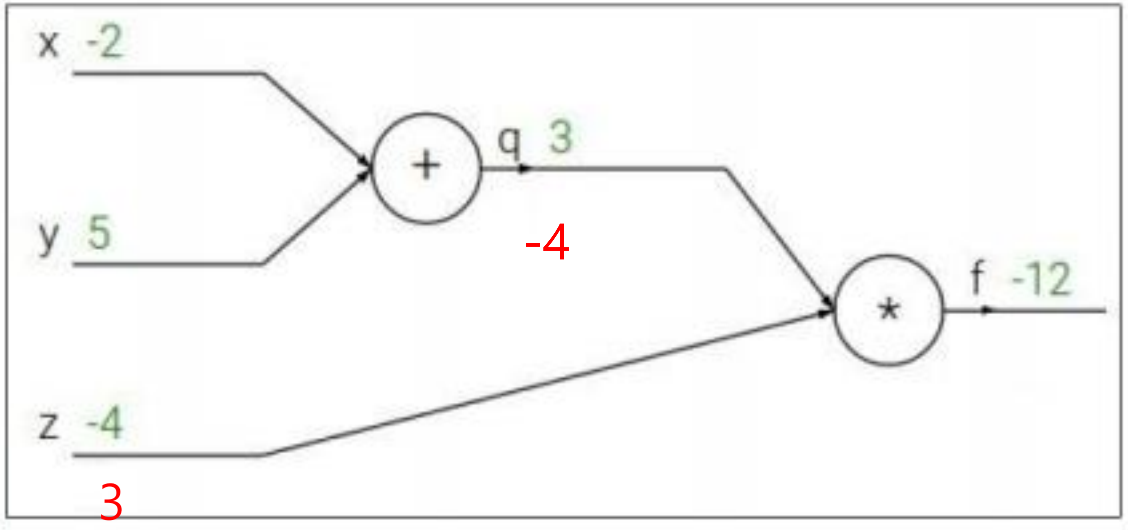
$$f(x,y,z) = (x+y)z$$

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



$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

$$\frac{\partial f}{\partial q} = z = -4$$

(출처 : cs231n_lecture4 p.13)

Forward & Back Prop.

Backpropagation: a simple example

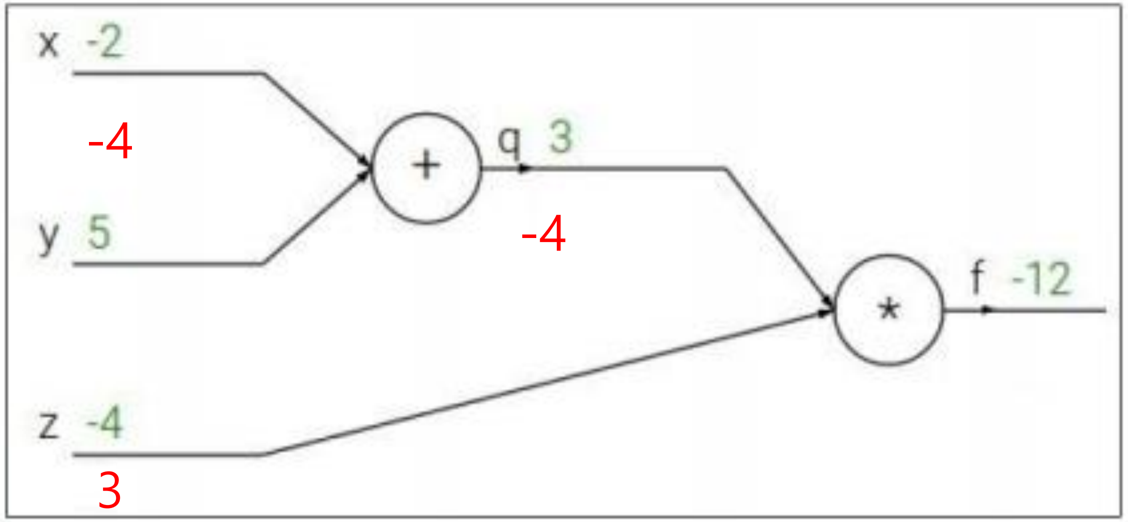
$$f(x, y, z) = (x + y)z$$

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



$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

$$\frac{\partial f}{\partial q} = z = -4 \;=\; \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} \; -4 * 1 = -4$$

(출처 : cs231n_lecture4 p.13)

Forward & Back Prop.

Backpropagation: a simple example

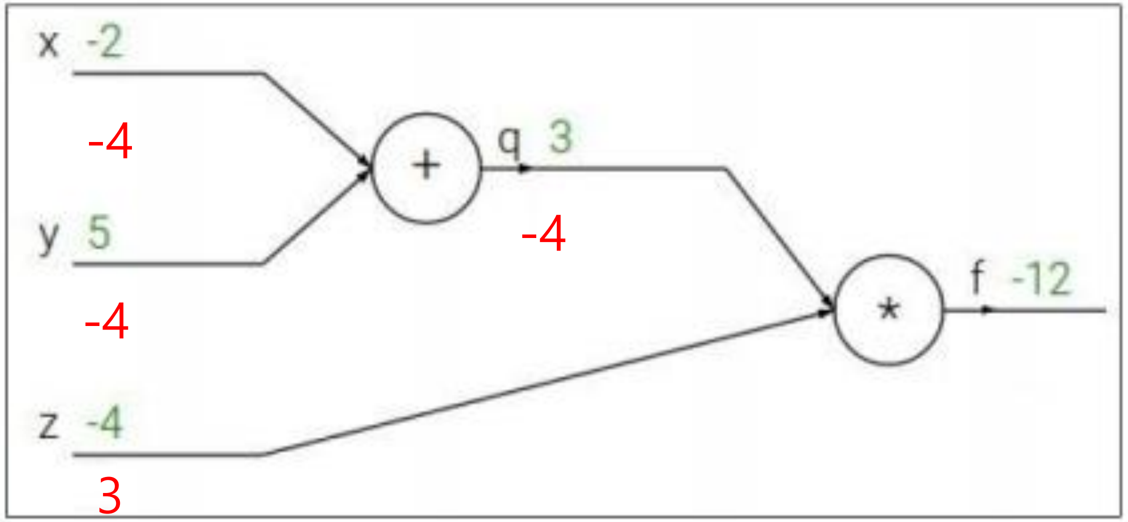
$$f(x,y,z) = (x+y)z$$

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



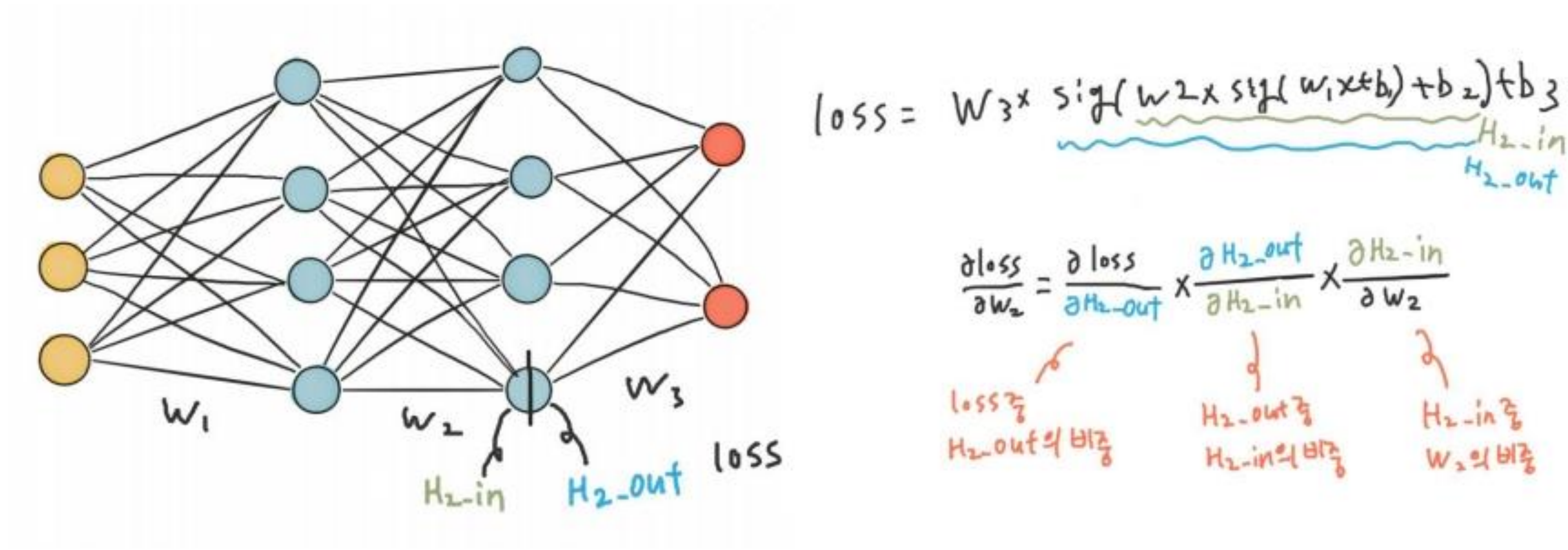
$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

$$\frac{\partial f}{\partial q} = z = -4 \; = \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} \; -4 * 1 = -4$$

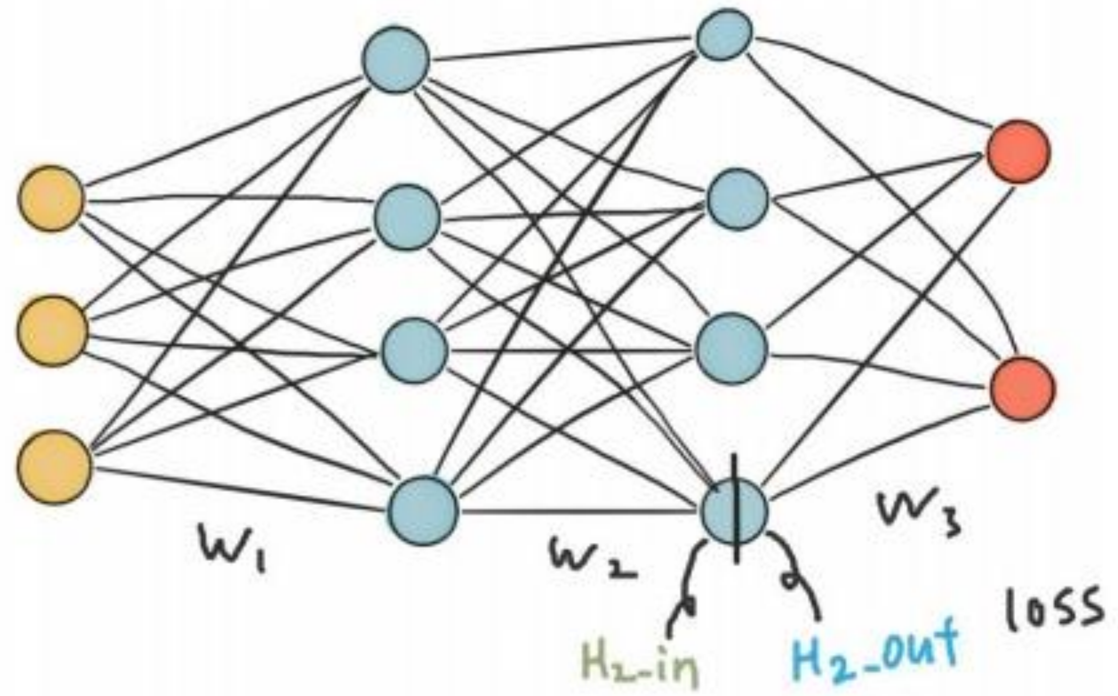
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = -4 * 1 = -4$$

(출처 : cs231n_lecture4 p.13)

Forward & Back Prop.



Forward & Back Prop.



$$loss = W_3 \times sig(w_2 \times sig(w_1 x + b_1) + b_2) + b_3 - y$$

H_2_in

H_2_out

$$\frac{\partial loss}{\partial w_2} = \frac{\partial loss}{\partial H_2\_out} \times \frac{\partial H_2\_out}{\partial H_2\_in} \times \frac{\partial H_2\_in}{\partial w_2}$$

loss중 H_2_out의 비중

H_2_out중 H_2_in의 비중

H_2_in중 W_2의 비중

$$\frac{\partial loss}{\partial w2} = w3 * sigmoid^{'}(h2\_in) * sigmoid(w1 * x + b)$$

Forward & Back Prop.

(참고) sigmoid 함수의 미분

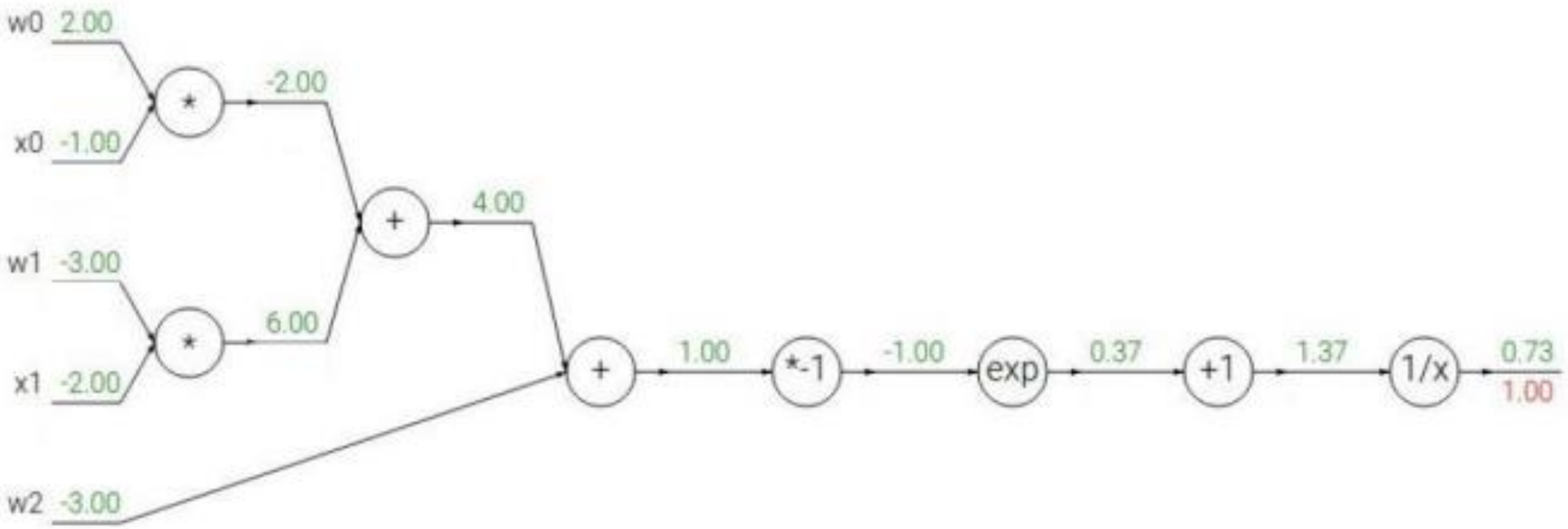
$$\sigma(x)' = \frac{\delta\{1+e^{-x}\}^{-1}}{\delta x} = -(1+e^{-x})^{-2}\text{-}e^{-x} = \frac{e^{-x}}{(1+e^{-x})^2}$$

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

(출처 : http://newsight.tistory.com/68)

Forward & Back Prop.

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



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

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

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

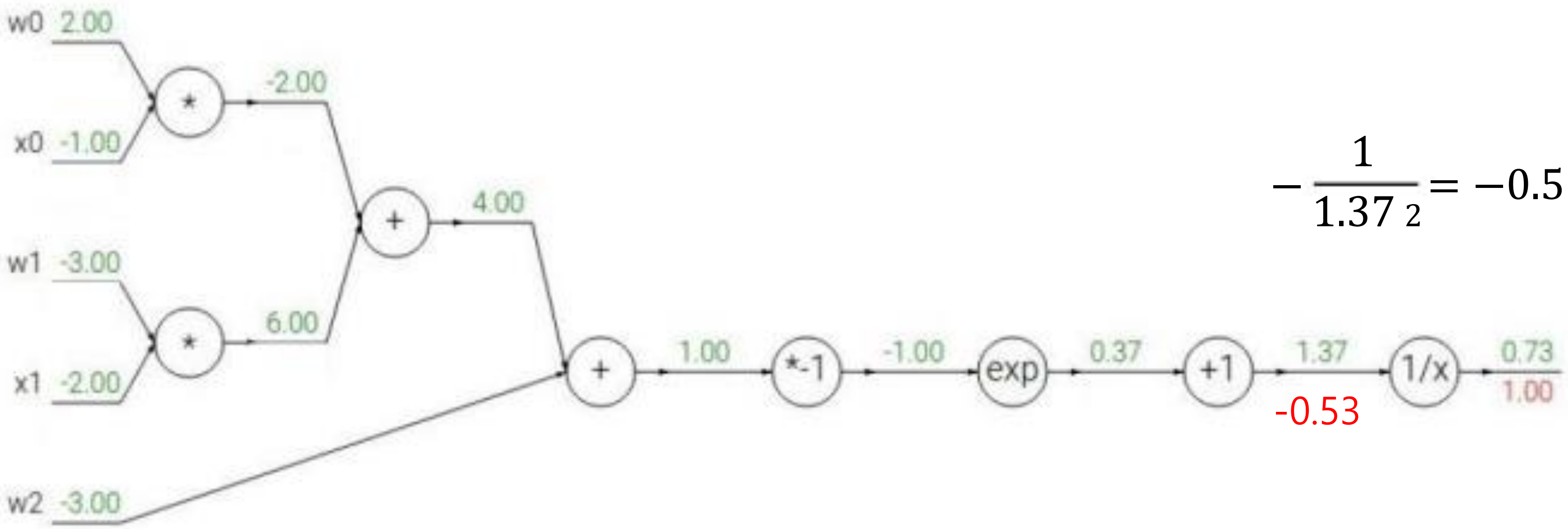
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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

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



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

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

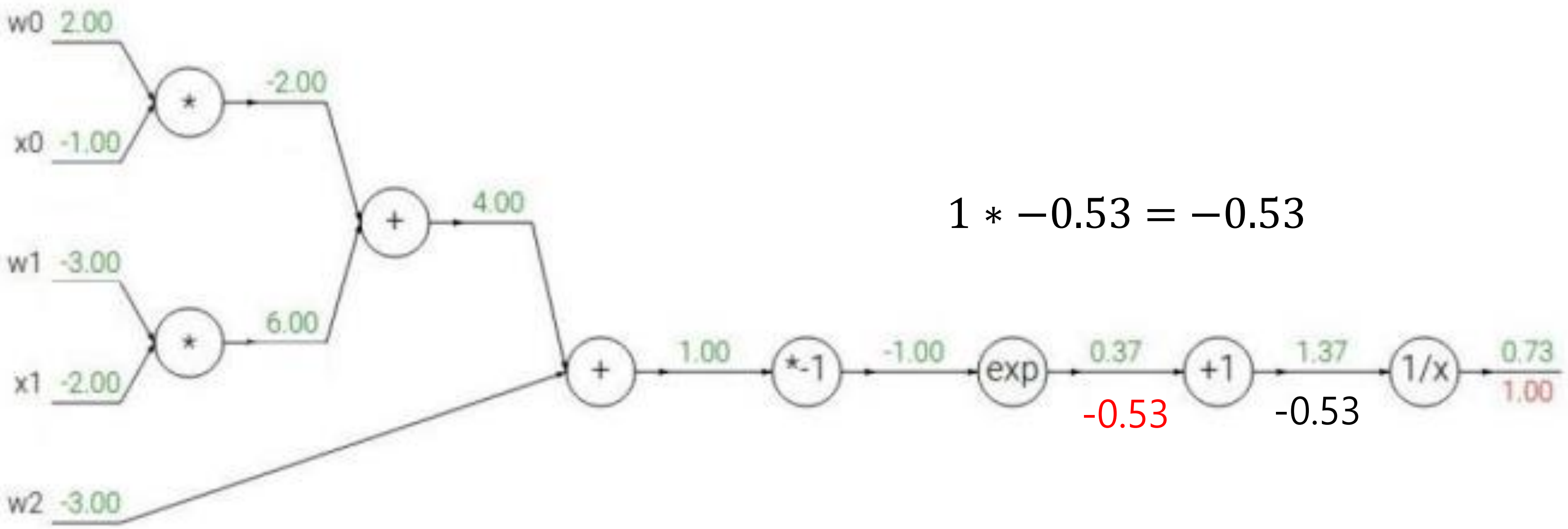
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



$1 * -0.53 = -0.53$

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

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

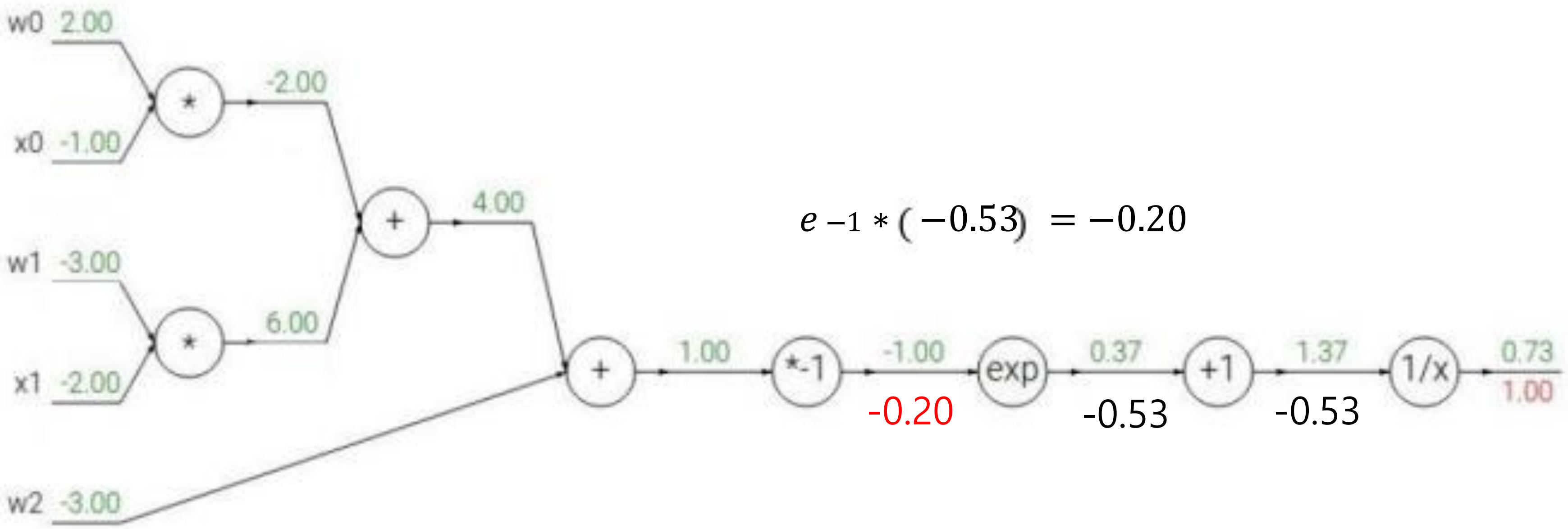
$f(x) = \dfrac{1}{x} \quad \rightarrow \quad \dfrac{df}{dx} = -1/x^2$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

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

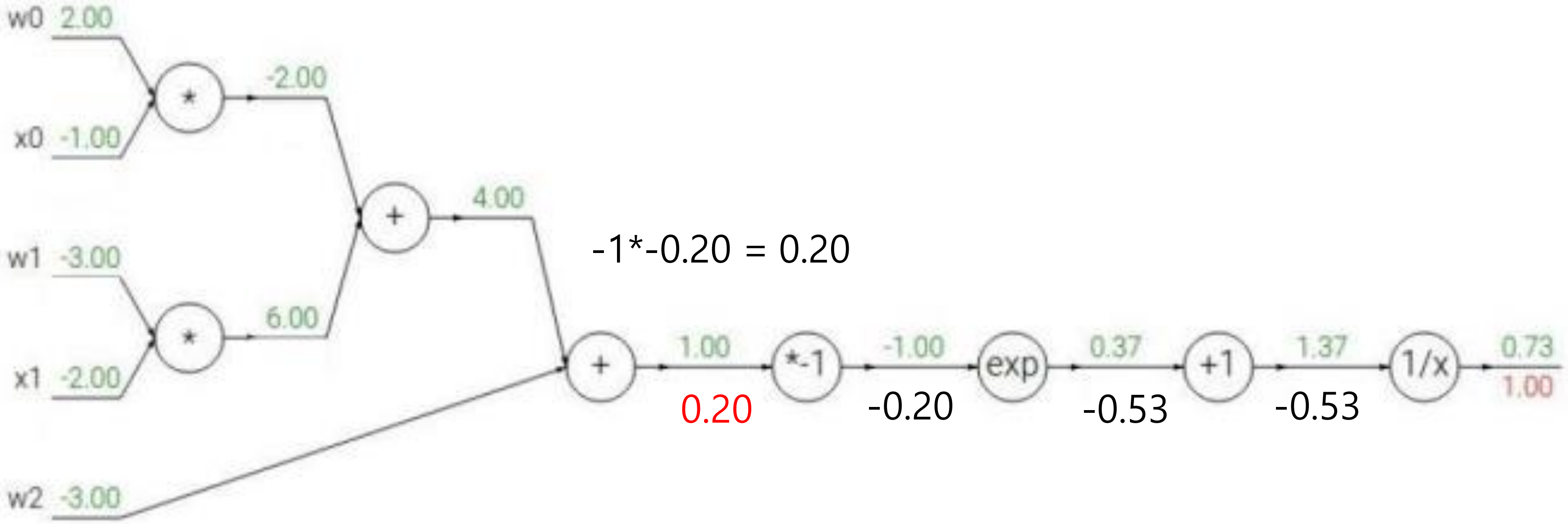
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

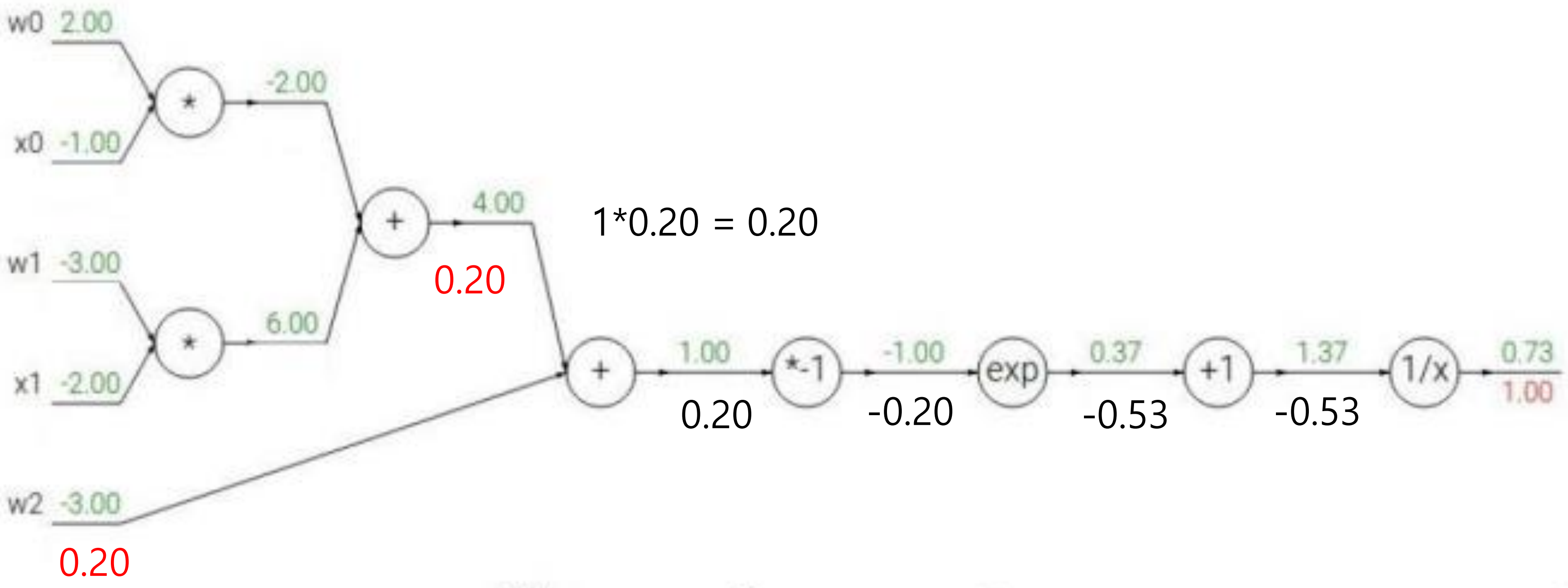
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



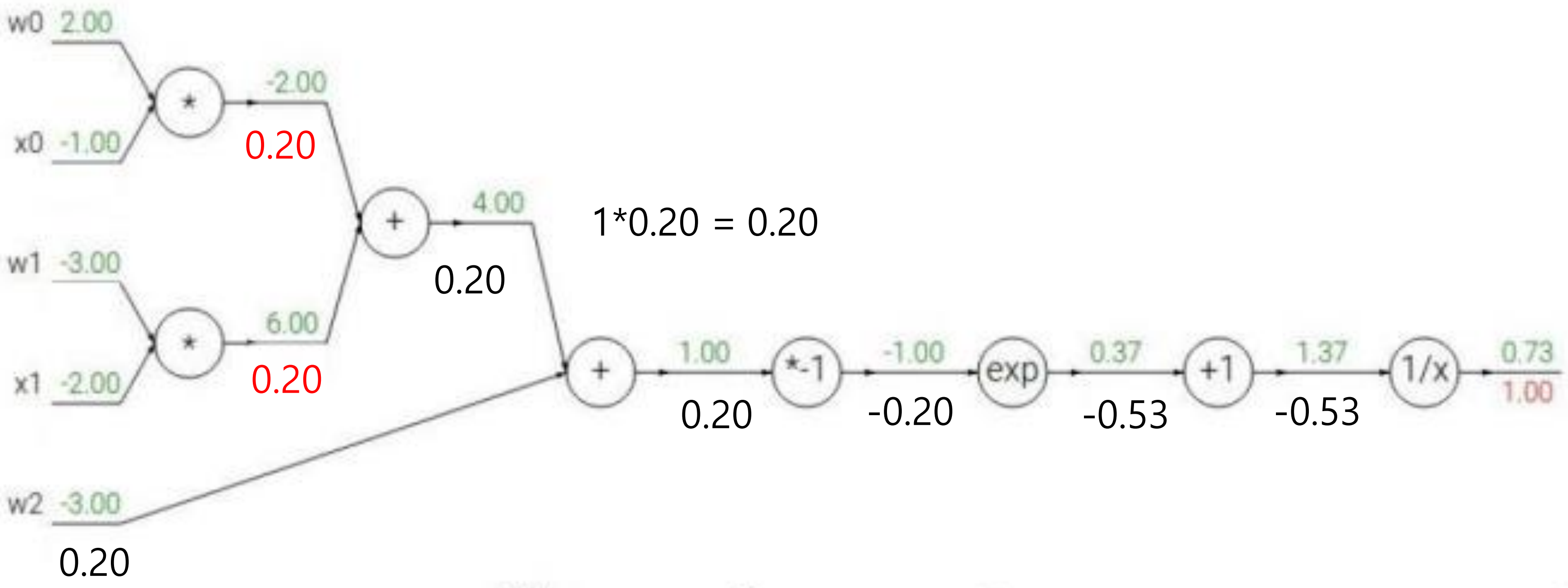
$f(x) = e^x \quad \rightarrow \quad \dfrac{df}{dx} = e^x$ | $f(x) = \dfrac{1}{x} \quad \rightarrow \quad \dfrac{df}{dx} = -1/x^2$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

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

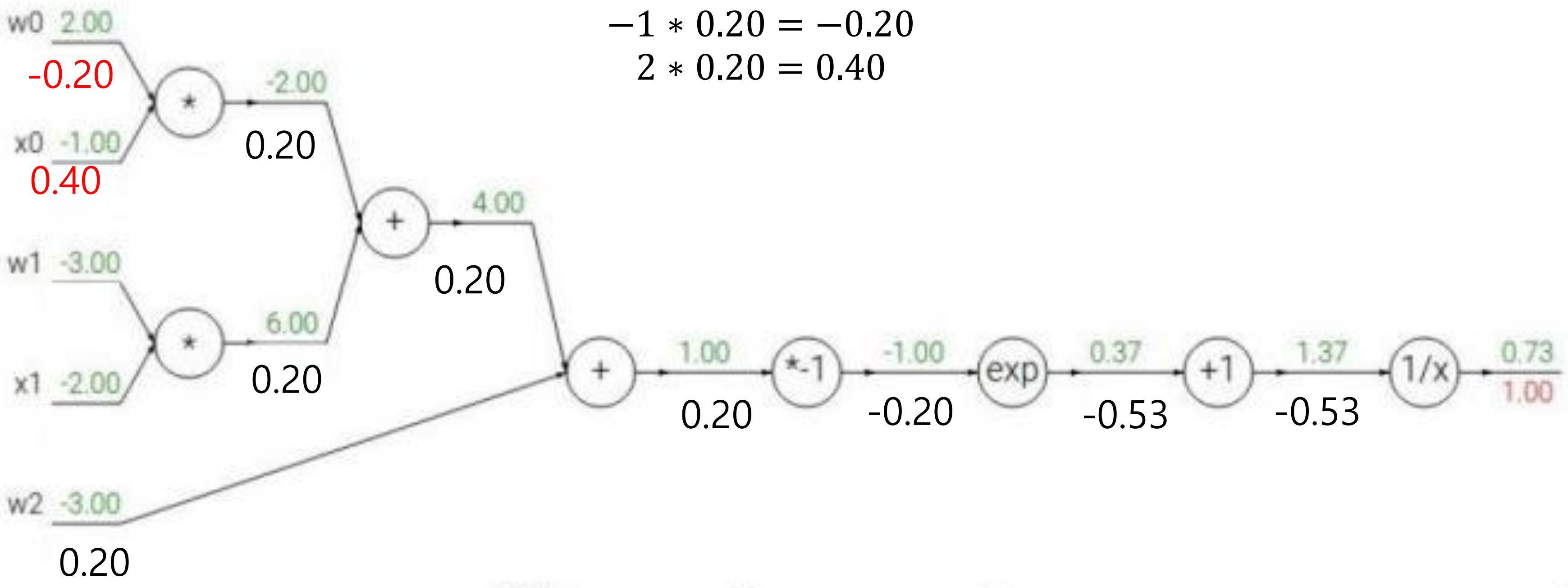
$f_c(x) = c + x \quad \rightarrow \quad \dfrac{df}{dx} = 1$

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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

$$-1 * 0.20 = -0.20$$
$$2 * 0.20 = 0.40$$



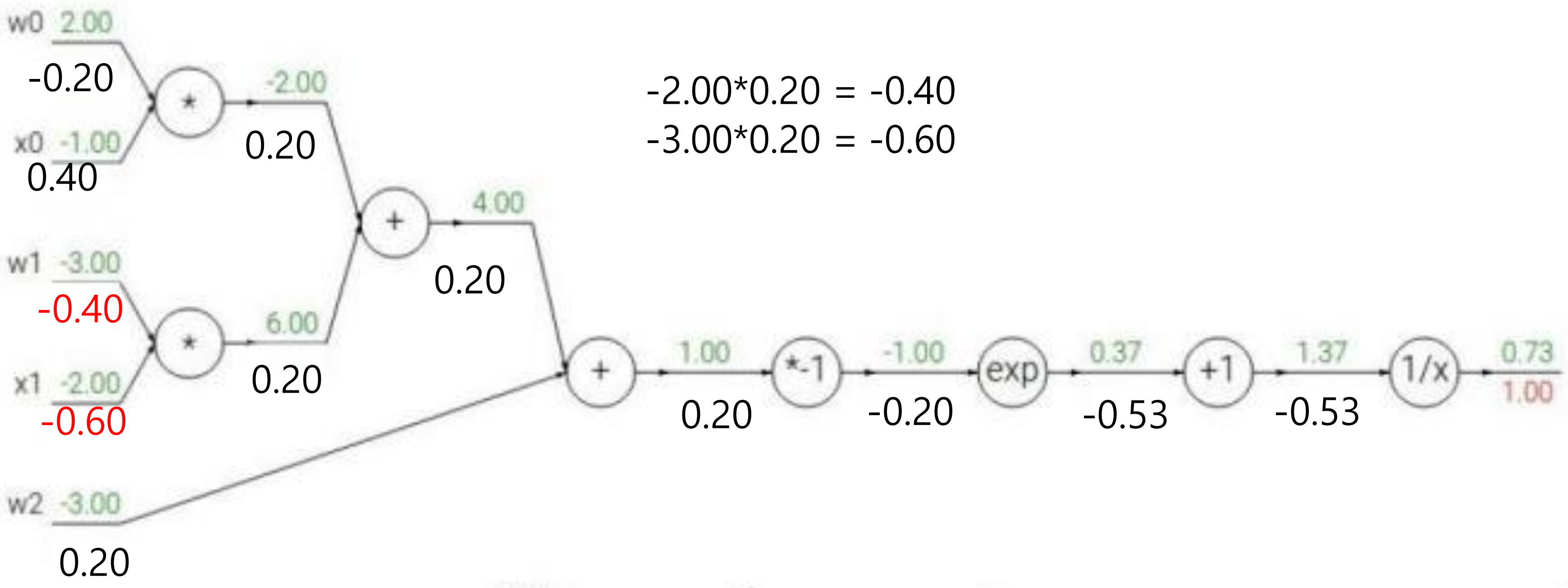
$f(x) = e^x \quad \rightarrow \quad \dfrac{df}{dx} = e^x$ | $f(x) = \dfrac{1}{x} \quad \rightarrow \quad \dfrac{df}{dx} = -1/x^2$

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

(출처 : cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

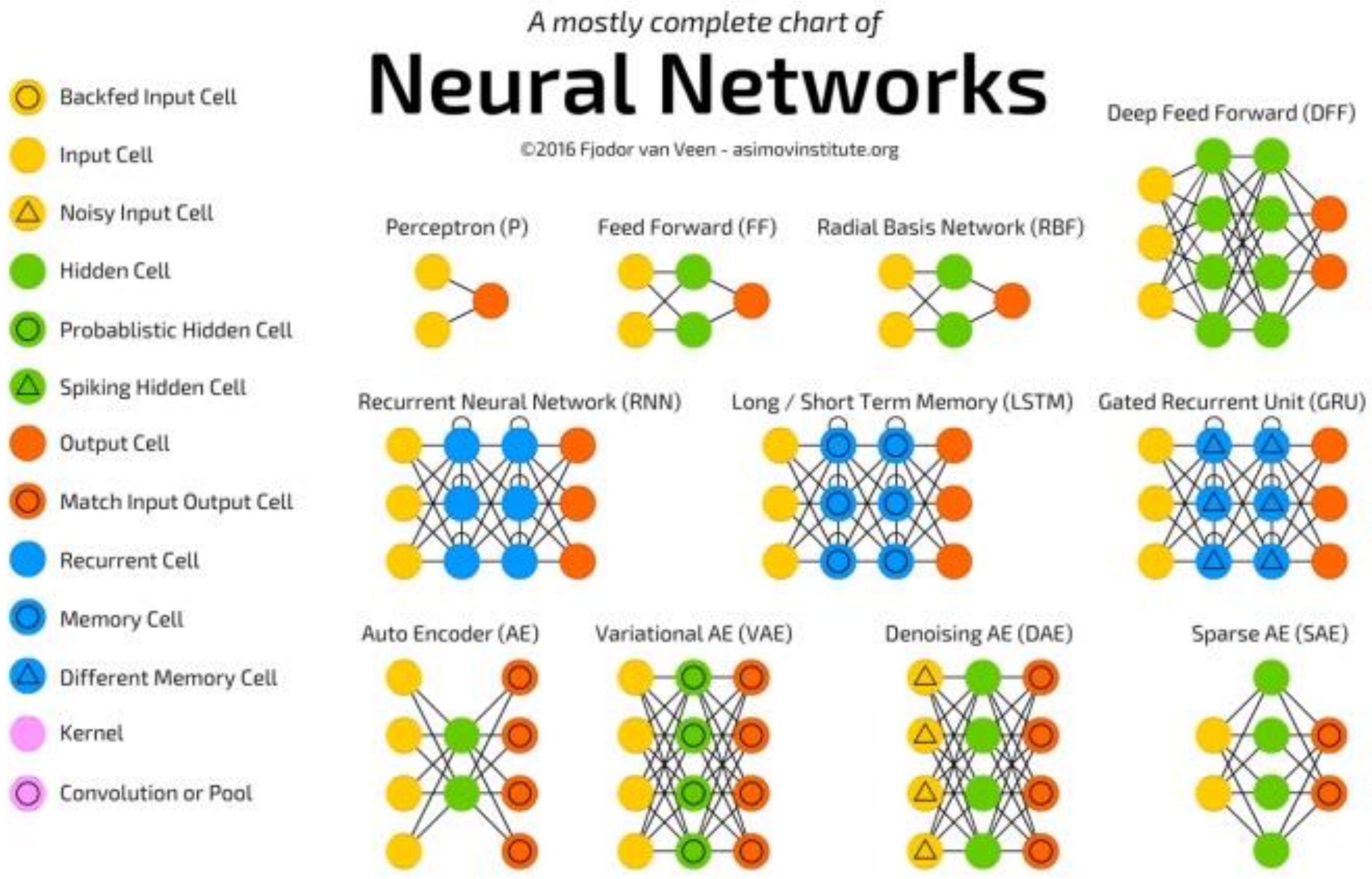
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

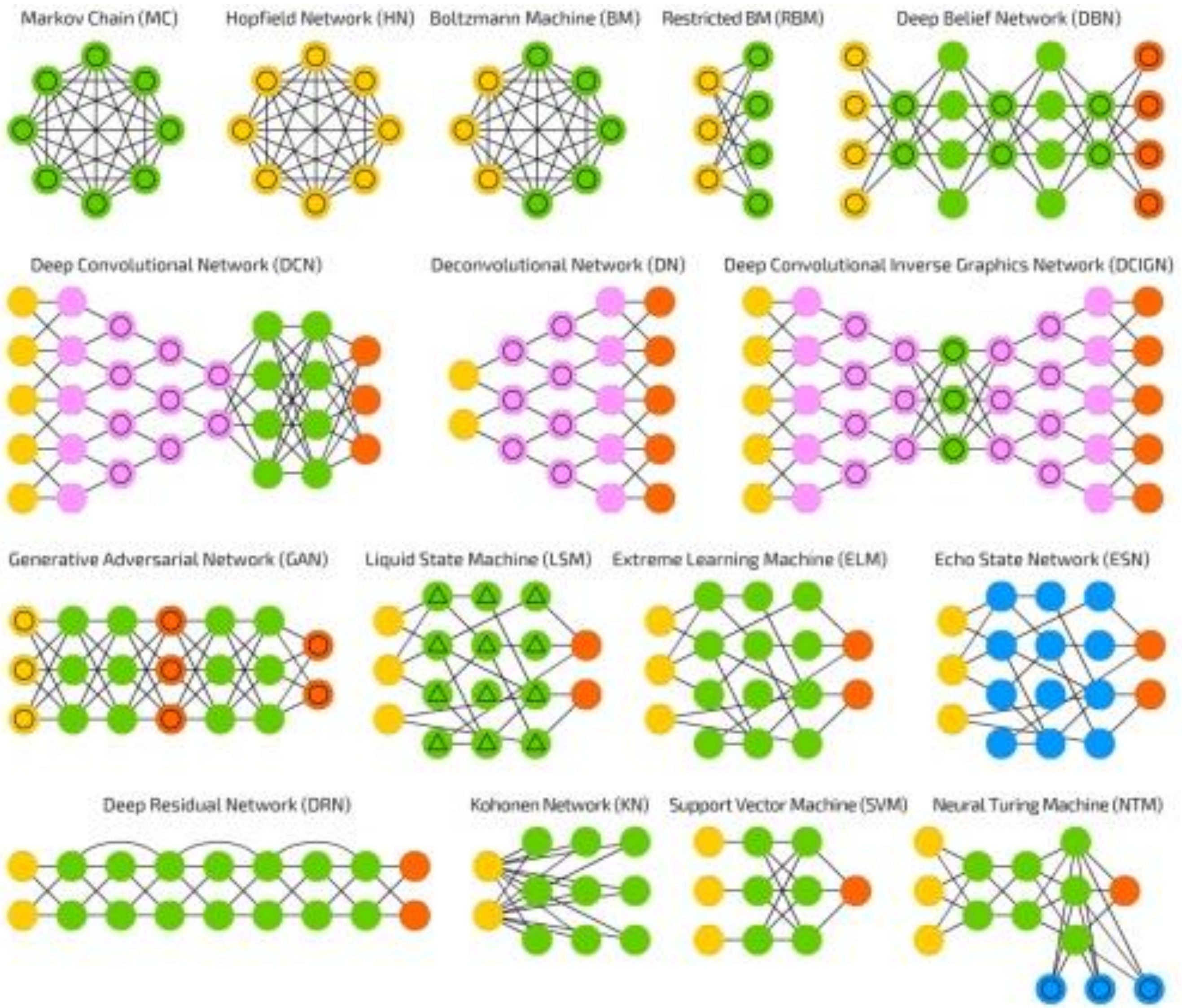
(출처 : cs231n_lecture4 p.31)

Neural Network



(출처 : http://www.asimovinstitute.org/neural-network-zoo/)

Neural Network



(출처 : http://www.asimovinstitute.org/neural-network-zoo/)

Forward & Back Prop.

$$Y = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}} = [1+e^{-(w_0x_0+w_1x_1+w_2)}]^{-1}$$

$w_0 = 2$
$w_1 = -3$
$w_2 = -3$
$x_0 = -1$
$x_1 = -2$

$$loss = \overset{\approx 0.73}{\hat{Y}} - Y$$

$$= \Big[1+\underbrace{e^{\overbrace{-(w_0x_0+w_1x_1+w_2)}^{\alpha=-1}}}_{\beta = 0.37}\Big]^{-1} - Y \quad \gamma = 1.37$$

$$\frac{\partial loss}{\partial \gamma} = (\gamma^{-1})' = -\frac{1}{\gamma^2} = -0.5327$$

$$\frac{\partial \gamma}{\partial \beta} = 1$$

$$\frac{\partial \beta}{\partial \alpha} = (e^{\alpha})' = e^{-1} = 0.3678$$

$$\frac{\partial \alpha}{\partial w_0} = -x_0 = 1$$

$loss = \gamma^{-1} - Y$
$\gamma = 1 + \beta$
$\beta = e^{\alpha}$

$$\frac{\partial loss}{\partial w_0} = \frac{\partial loss}{\partial \gamma} \times \frac{\partial \gamma}{\partial \beta} \times \frac{\partial \beta}{\partial \alpha} \times \frac{\partial \alpha}{\partial w_0}$$

$$= -0.5327 \times 1 \times 0.3678 \times 1$$

$$= -0.1959 \approx -0.2$$

Forward & Back Prop.

```
test.py
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
from torch.autograd import Variable
from visdom import Visdom
viz = Visdom()

num_data = 1000
num_epoch = 5000

x = init.uniform(torch.Tensor(num_data,1),-15,15)
y = 8*(x**2) + 7*x + 3

noise = init.normal(torch.FloatTensor(num_data,1),std=1)
y_noise = y + noise
```

Forward & Back Prop.

필요한 라이브러리

```python
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
from torch.autograd import Variable
from visdom import Visdom
viz = Visdom()

num_data = 1000
num_epoch = 5000

x = init.uniform(torch.Tensor(num_data,1),-15,15)
y = 8*(x**2) + 7*x + 3

noise = init.normal(torch.FloatTensor(num_data,1),std=1)
y_noise = y + noise
```

Forward & Back Prop.

필요한 라이브러리

데이터 생성

test.py

```python
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
from torch.autograd import Variable
from visdom import Visdom
viz = Visdom()

num_data = 1000
num_epoch = 5000

x = init.uniform(torch.Tensor(num_data,1),-15,15)
y = 8*(x**2) + 7*x + 3

noise = init.normal(torch.FloatTensor(num_data,1),std=1)
y_noise = y + noise
```

Forward & Back Prop.

```
model = nn.Sequential(
            nn.Linear(1,10),
            nn.ReLU(),
            nn.Linear(10,6),
            nn.ReLU(),
            nn.Linear(6,1),
      ).cuda()

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.001)

loss_arr =[]
label = Variable(y_noise.cuda())
for i in range(num_epoch):
      output = model(Variable(x.cuda()))
      optimizer.zero_grad()

      loss = loss_func(output,label)
      loss.backward()
      optimizer.step()
      if i % 100 ==0:
            print(loss)
      loss_arr.append(loss.cpu().data.numpy()[0])

param_list = list(model.parameters())
print(param_list)
```

Forward & Back Prop.

Neural Network 모델 생성

loss function 및
gradient descent
optimizer 생성

```python
model = nn.Sequential(
            nn.Linear(1,10),
            nn.ReLU(),
            nn.Linear(10,6),
            nn.ReLU(),
            nn.Linear(6,1),
        ).cuda()

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.001)

loss_arr =[]
label = Variable(y_noise.cuda())
for i in range(num_epoch):
    output = model(Variable(x.cuda()))
    optimizer.zero_grad()

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 100 ==0:
        print(loss)
    loss_arr.append(loss.cpu().data.numpy()[0])

param_list = list(model.parameters())
print(param_list)
```

Forward & Back Prop.

Neural Network 모델 생성

loss function 및
gradient descent
optimizer 생성

```python
model = nn.Sequential(
            nn.Linear(1,10),
            nn.ReLU(),
            nn.Linear(10,6),
            nn.ReLU(),
            nn.Linear(6,1),
        ).cuda()

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.001)

loss_arr =[]
label = Variable(y_noise.cuda())
for i in range(num_epoch):
    output = model(Variable(x.cuda()))
    optimizer.zero_grad()

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 100 ==0:
        print(loss)
    loss_arr.append(loss.cpu().data.numpy()[0])

param_list = list(model.parameters())
print(param_list)
```

Forward & Back Prop.

Neural Network 모델 생성

loss function 및
gradient descent
optimizer 생성

<training 단계 >

1. 모델로 결과값 추정
2. loss 및 gradient 계산
3. 모델 업데이트

```
model = nn.Sequential(
            nn.Linear(1,10),
            nn.ReLU(),
            nn.Linear(10,6),
            nn.ReLU(),
            nn.Linear(6,1),
       ).cuda()

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.001)

loss_arr =[]
label = Variable(y_noise.cuda())
for i in range(num_epoch):
    output = model(Variable(x.cuda()))
    optimizer.zero_grad()

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 100 ==0:
        print(loss)
    loss_arr.append(loss.cpu().data.numpy()[0])

param_list = list(model.parameters())
print(param_list)
```

Forward & Back Prop.

Neural Network 모델 생성

loss function 및
gradient descent
optimizer 생성

<training 단계 >

1. 모델로 결과값 추정
2. loss 및 gradient 계산
3. 모델 업데이트

training 이후 파라미터 값 확인

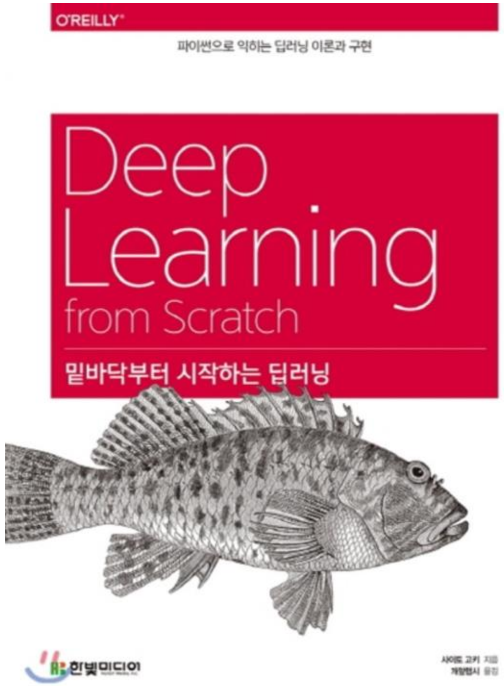
```python
model = nn.Sequential(
            nn.Linear(1,10),
            nn.ReLU(),
            nn.Linear(10,6),
            nn.ReLU(),
            nn.Linear(6,1),
      ).cuda()

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.001)

loss_arr =[]
label = Variable(y_noise.cuda())
for i in range(num_epoch):
    output = model(Variable(x.cuda()))
    optimizer.zero_grad()

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 100 ==0:
        print(loss)
    loss_arr.append(loss.cpu().data.numpy()[0])

param_list = list(model.parameters())
print(param_list)
```

퍼셉트론의 학습과정

1. 임의의 w와 b를 설정
2. 주어진 훈련 데이터를 이용하여, 결과값(y)를 도출
3. 결과값(y)와 실제 결과값($\hat{y}$) 사이의 오차 계산
4. 오차(Loss)를 줄이는 방향으로 w와 b를 재설정(학습)
5. 오차를 최소한으로 줄이도록 2~4의 과정을 계속반복진행

Book

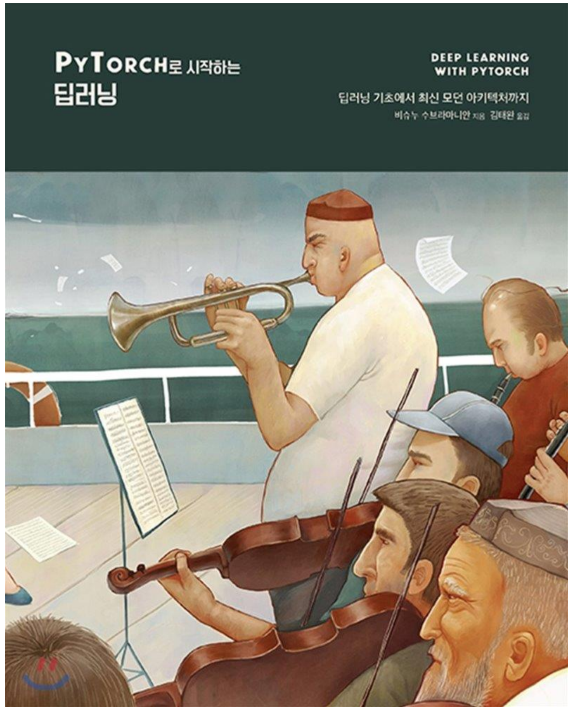
주교재



밑바닥부터 시작하는 딥러닝 1, 사이토 고키 지음, 개앞맨시 옮김 .

(Deep Learning from Scratch의 번역서입니다 .)

부교재



Pytorch로 시작하는 딥러닝, 비슈누 수브라마니안 지음 김태완 옮김 .

(Deep Learning with PYTORCH 의 번역서입니다 .)