Linear Regression & Neural Network

2017.10.21

최건호

INDEX

01

Linear Regression

- 정의
- Loss
- Gradient Descent
- 문제점

02

Neural Network

- 정의
- Deep?
- 활용

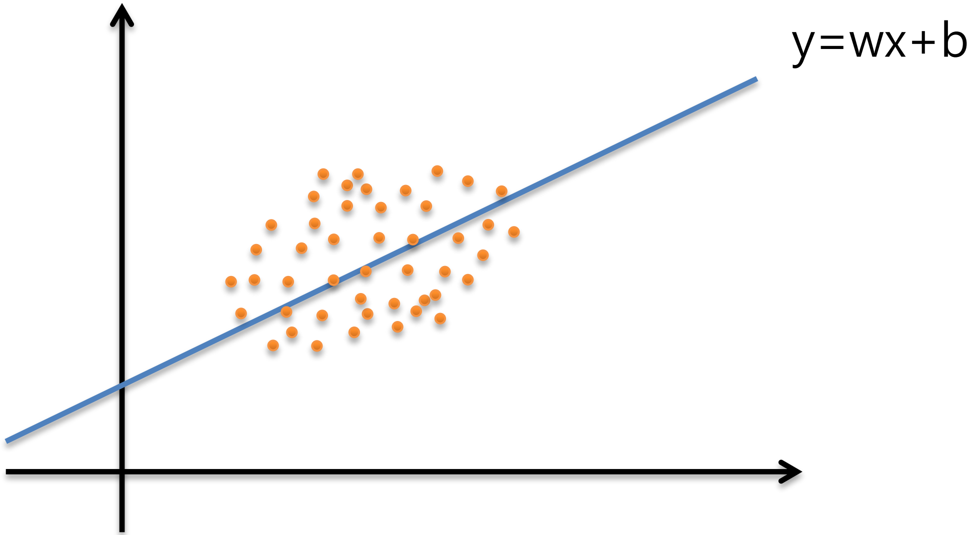
03

Propagation

- 정의
- Forward Prop.
- Back Prop.

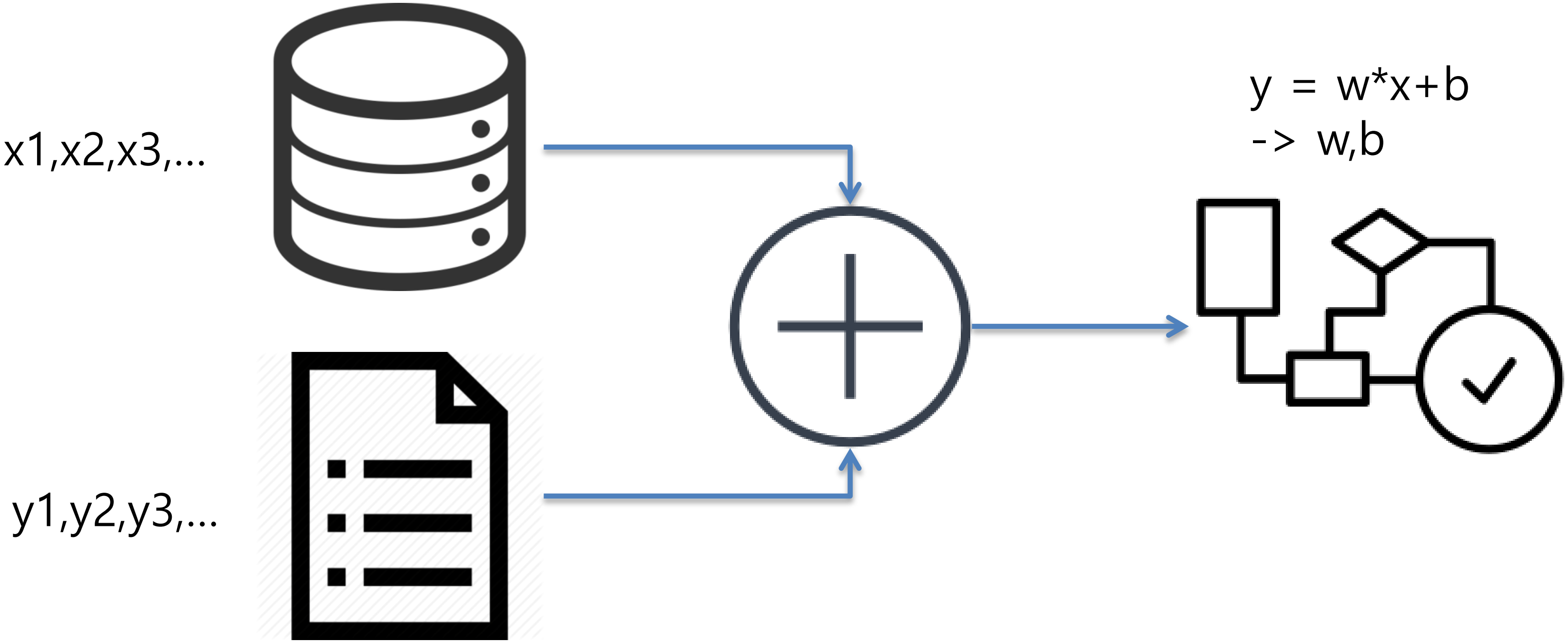
Linear Regression

x에 대한 y값을 가장 잘 설명하는
변수 w, b를 찾는 것



종속 변수 y와 한 개 이상의 독립 변수 X와의
선형 상관 관계를 모델링하는 회귀분석 기법
(출처: 위키피디아)

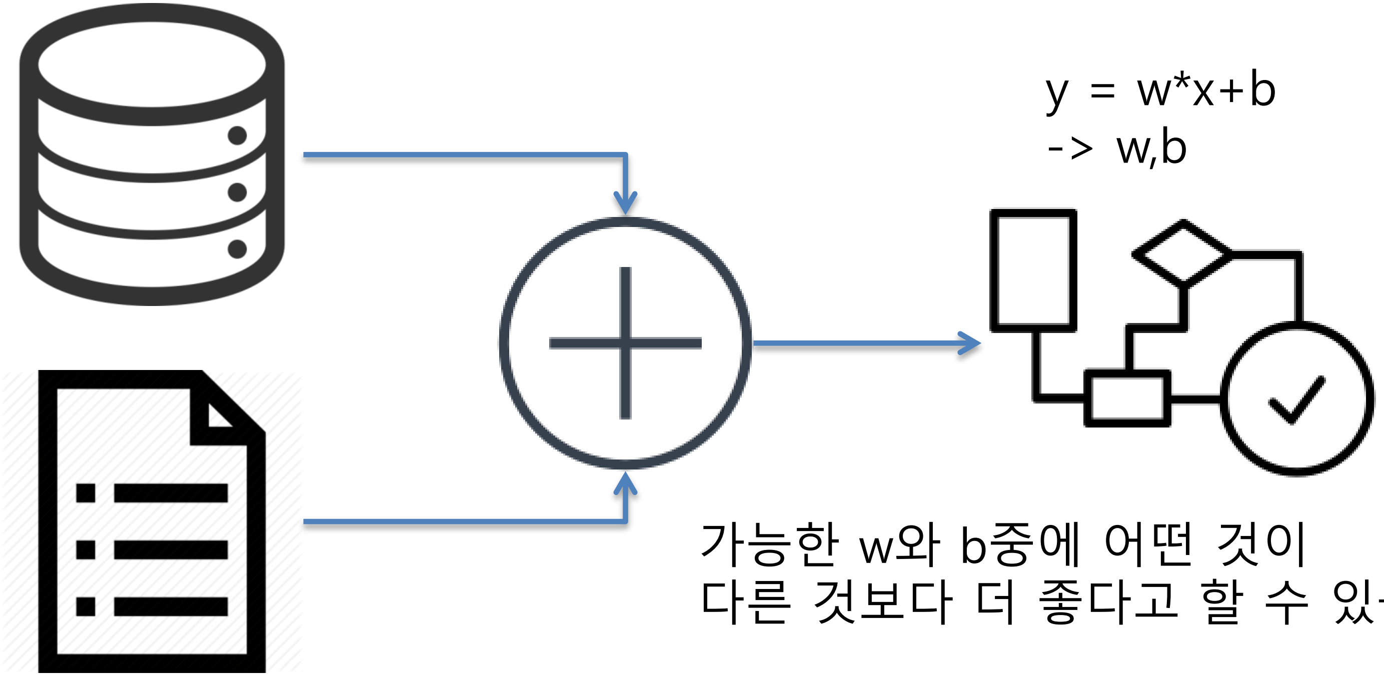
Linear Regression



Linear Regression

x1,x2,x3,...

y1,y2,y3,...



$y = w*x+b$

-> w,b

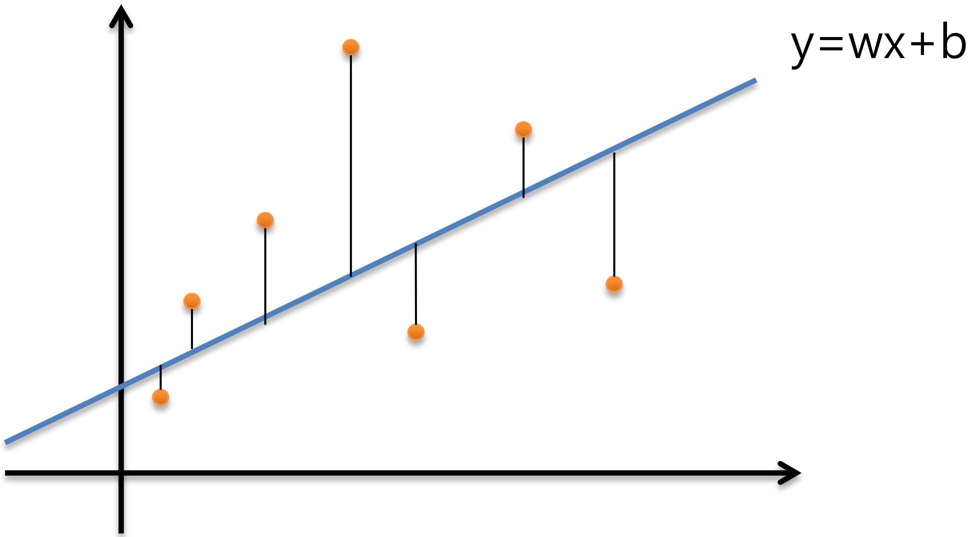
가능한 w와 b중에 어떤 것이
다른 것보다 더 좋다고 할 수 있을까?

Linear Regression

잘 예측했는지 아닌지 측정할 척도(metric)가 필요함

Linear Regression

잘 예측했는지 아닌지 측정할 척도(metric)가 필요함



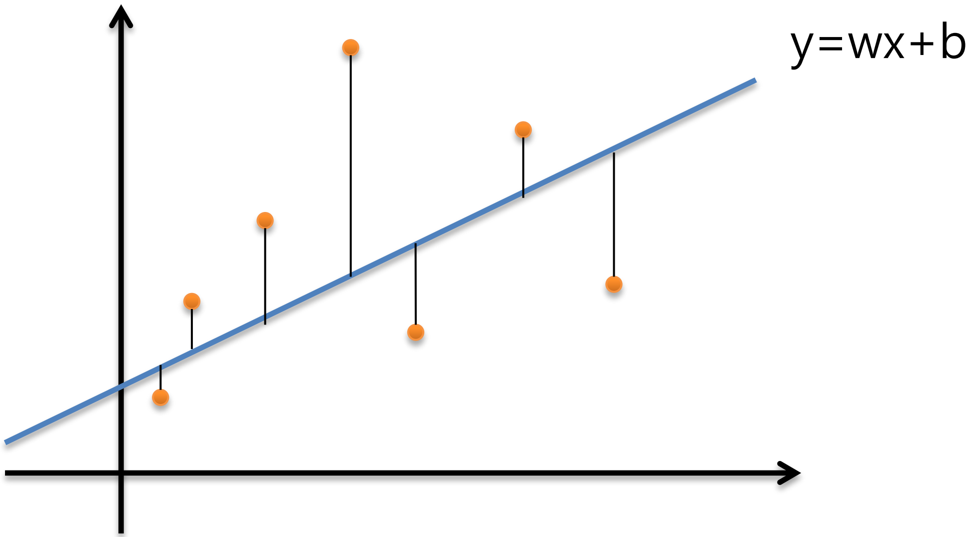
Linear Regression

Mean Squared Error(MSE)

$$MSE = \frac{(x1 - x2)^2}{n}$$

두 값의 거리의 제곱의 평균

잘 예측했는지 아닌지 측정할 척도(metric)가 필요함



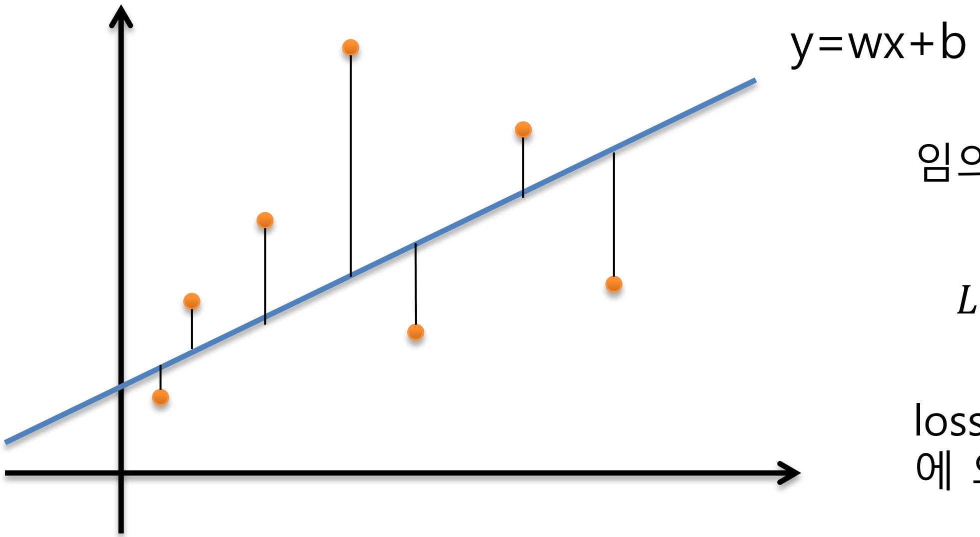
Linear Regression

Mean Squared Error(MSE)

$$MSE = \frac{(x1 - x2)^2}{n}$$

두 값의 거리의 제곱의 평균

잘 예측했는지 아닌지 측정할 <u>척도(metric)</u>가 필요함



임의의 w*,b*를 초기값으로 한다면

$$Loss = \frac{(y^* - y)^2}{n} = \frac{(w^*x + b^* - y)^2}{n}$$

loss값은 고정된 x, y(데이터)에서 w*,b* 에 의해 구해짐

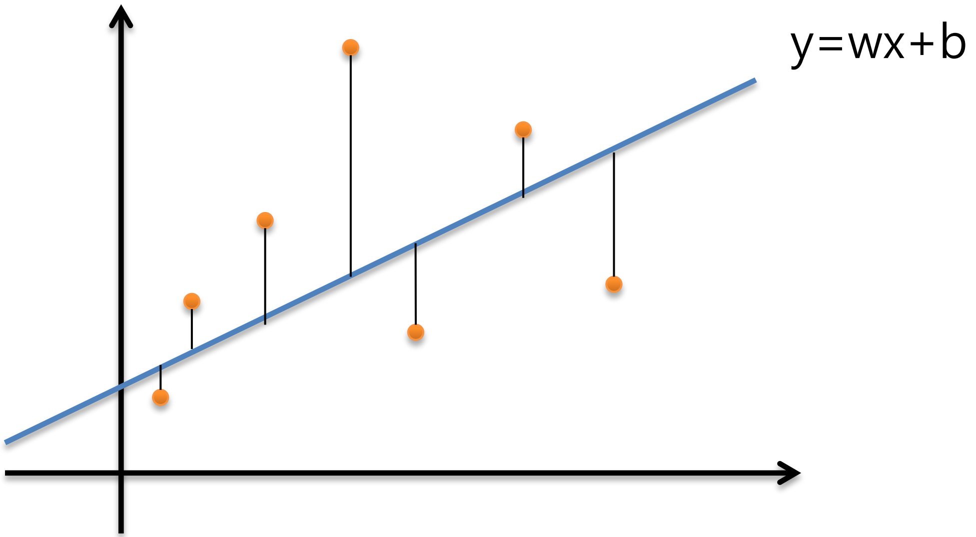
Linear Regression

Mean Squared Error(MSE)

$$MSE = \frac{(x1 - x2)^2}{n}$$

잘 예측했는지 아닌지 측정할 척도(metric)가 필요함

두 값의 거리의 제곱의 평균



예시)
y = 0.5*x+4 이고 w*=2,
b*=2 일 때, x=3에서 loss는?

$$Loss = \frac{(8 - 5.5)^2}{n} = \frac{(2.5)^2}{n}$$

Linear Regression

Loss를 minimize하는 w,b를 구하고 싶다.

Linear Regression

Loss를 minimize하는 w,b를 구하고 싶다.

-> Random Search?

Linear Regression

Loss를 minimize하는 w,b를 구하고 싶다.

-> Random Search? 어느 세월에..

Linear Regression

Loss를 minimize하는 w,b를 구하고 싶다.

-> Random Search? 어느 세월에..
-> Loss 값을 통해서 구할 수 없을까?

Linear Regression

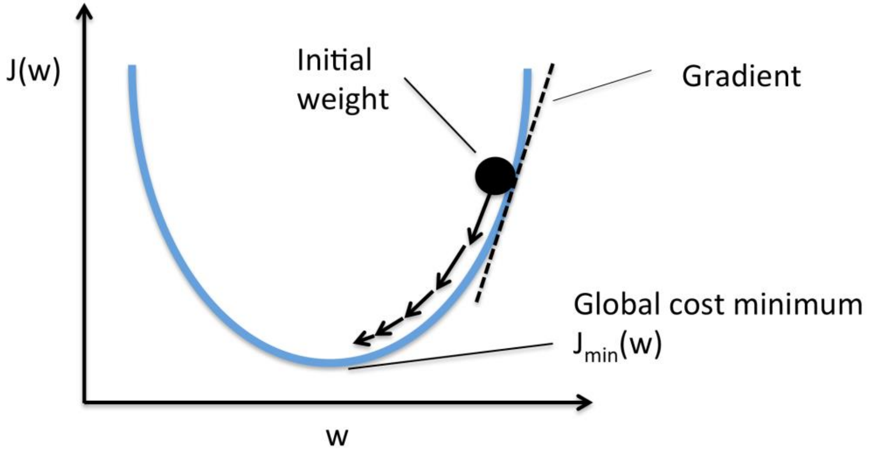
Loss를 minimize하는 w,b를 구하고 싶다.

-> Random Search? 어느 세월에..
-> Loss 값을 통해서 구할 수 없을까? Gradient Descent!

Linear Regression

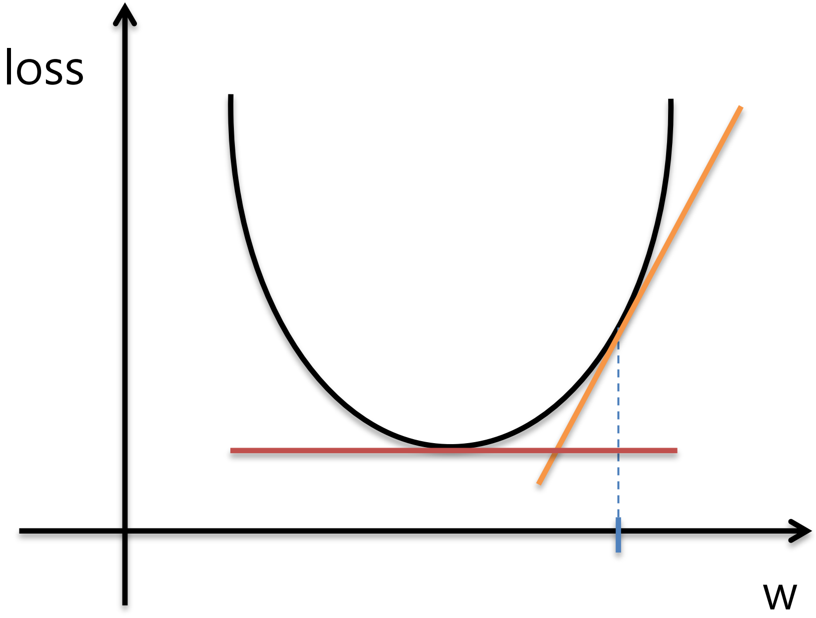
Loss를 minimize하는 w,b를 구하고 싶다.

-> Random Search? 어느 세월에..
-> Loss 값을 통해서 구할 수 없을까? Gradient Descent!



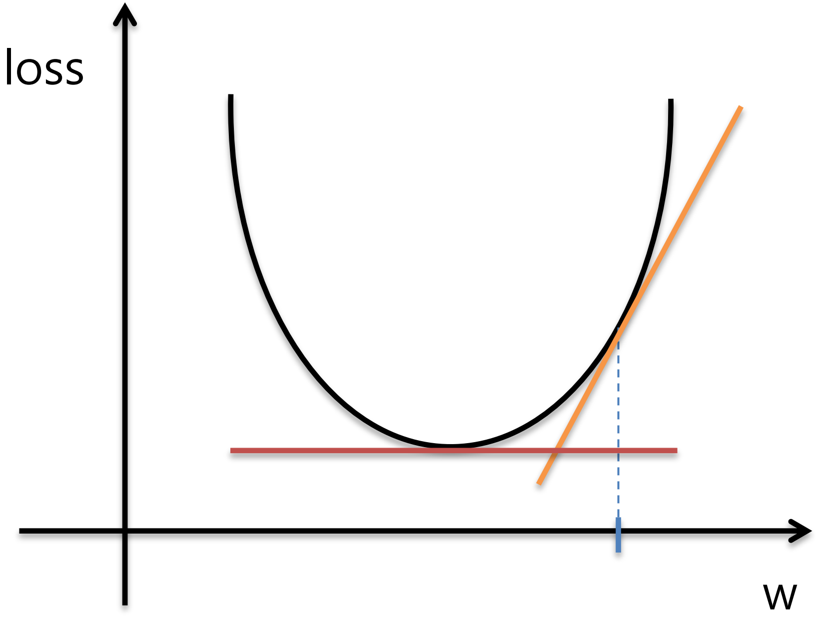
(출처: https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html)

Linear Regression



그냥 바로 loss를 w에 대해 미분 했을 때 그 값이 0이 되는 w를 찾으면 안될까?

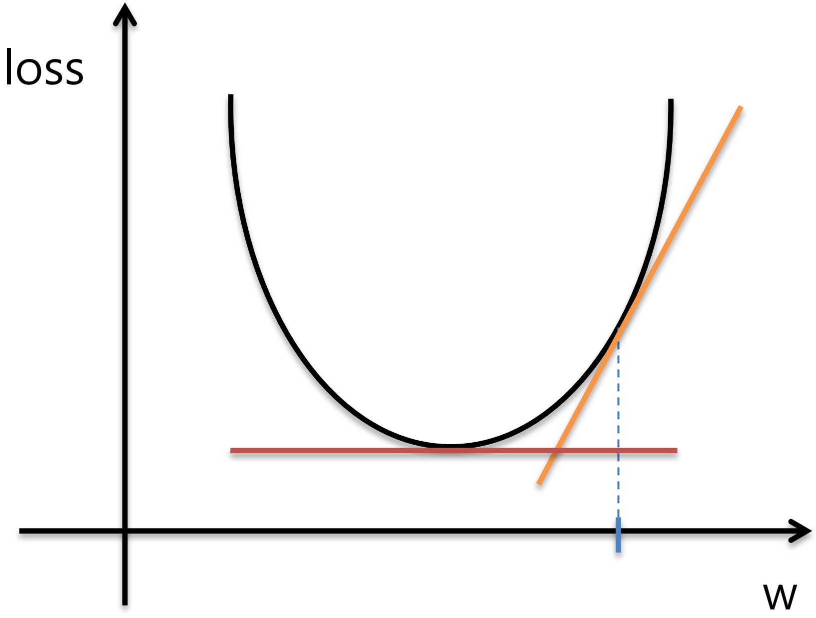
Linear Regression



그냥 바로 loss를 w에 대해 미분 했을 때 그 값이 0이 되는 w를 찾으면 안될까?

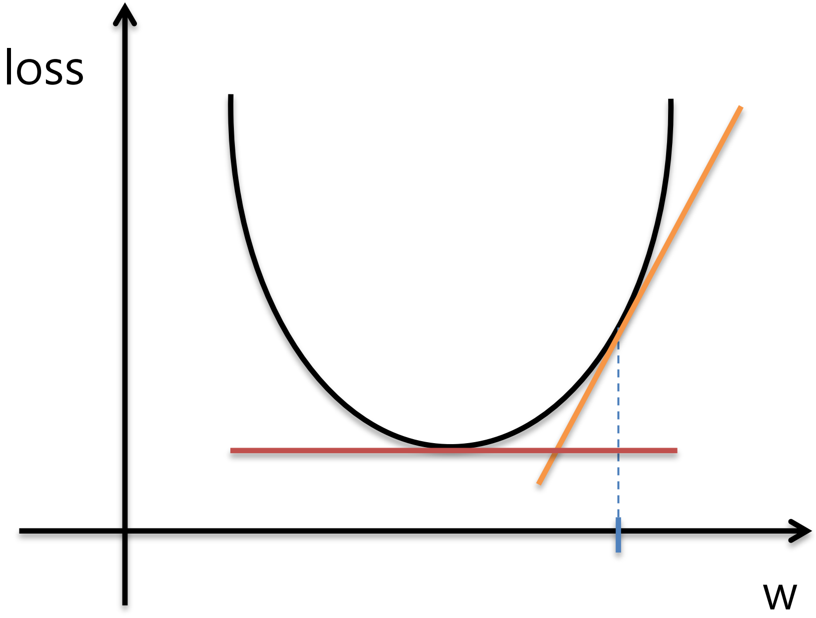
loss를 minimize하는 w를 구하려면 $w = (x^T x)^{-1} x^T y$ 식을 풀어야 하는데, 변수가 많아지고 matrix가 커지면 복잡도가 $O(n^3)$ 이기 때문에 exponential 하게 증가한다. 변수가 많아질수록 계산이 거의 불가능.

Linear Regression



바로 minimum이 되는 지점을 구하기보다, 현재 loss에 대한 w의 gradient(경사도)를 구하여 gradient x learning rate만큼 w를 업데이트

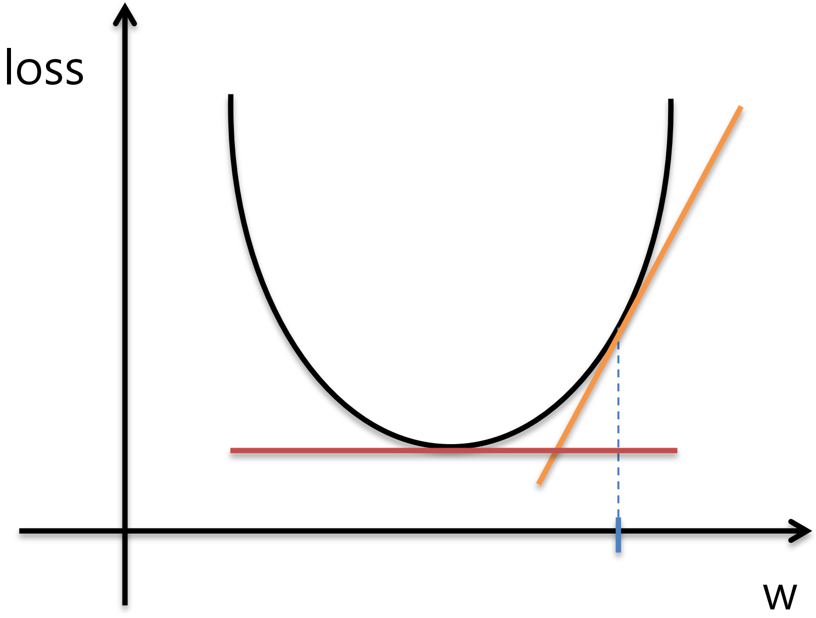
Linear Regression



바로 minimum이 되는 지점을 구하기보다, 현재 loss에 대한 w의 gradient(경사도)를 구하여 gradient x learning rate만큼 w를 업데이트

$$w_{t+1} = \mathrm{w_t} - \mathrm{gradient} * \mathrm{learning\_rate}$$

Linear Regression

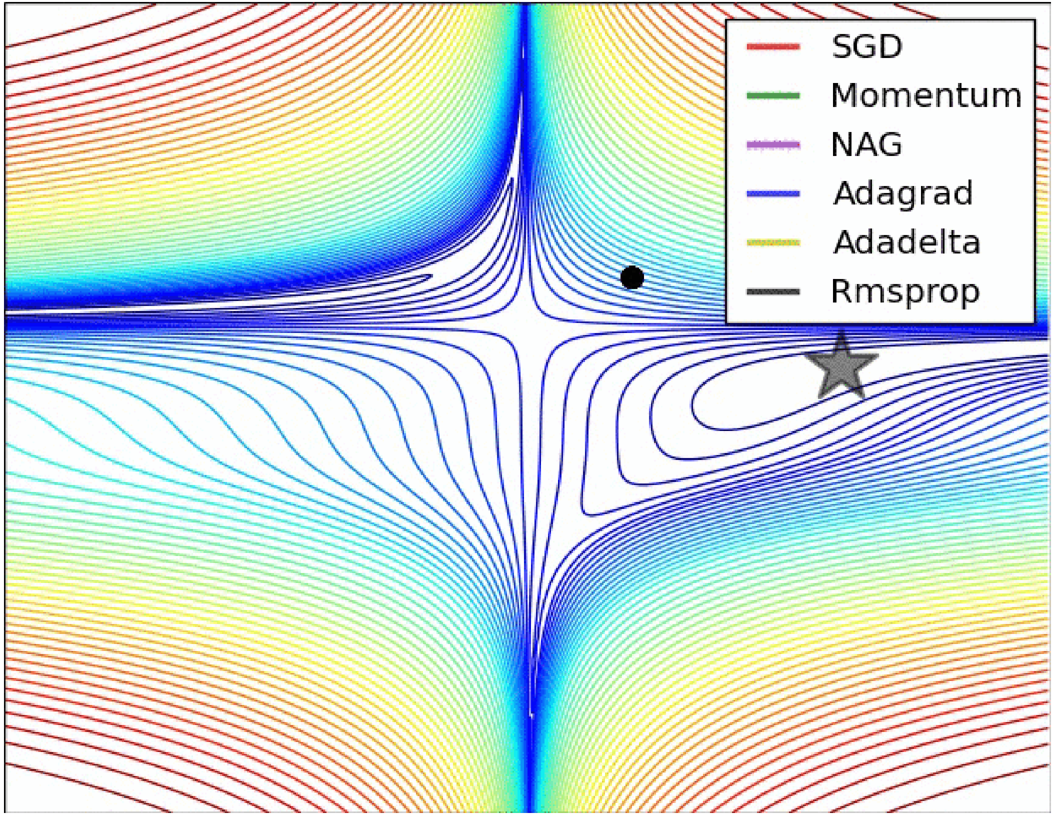


바로 minimum이 되는 지점을 구하기보다, 현재 loss에 대한 w의 gradient(경사도)를 구하여 gradient x learning rate만큼 w를 업데이트

$$w_{t+1} = \mathrm{w_t} - \mathrm{gradient} * \mathrm{learning\_rate}$$

계속 업데이트 하다 보면 optimal 한 w에 근접하게 된다.

Linear Regression



(출처: http://imgur.com/a/Hqolp)

Linear Regression

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 = 1000

noise = init.normal(torch.FloatTensor(num_data,1),std=0.2)
x = init.uniform(torch.Tensor(num_data,1),-10,10)
y = 2*x+3
y_noise = y + noise
```

Linear Regression

필요한 라이브러리

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 = 1000

noise = init.normal(torch.FloatTensor(num_data,1),std=0.2)
x = init.uniform(torch.Tensor(num_data,1),-10,10)
y = 2*x+3
y_noise = y + noise
```

Linear Regression

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 = 1000

noise = init.normal(torch.FloatTensor(num_data,1),std=0.2)
x = init.uniform(torch.Tensor(num_data,1),-10,10)
y = 2*x+3
y_noise = y + noise
```

Linear Regression

```python
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.01)

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

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 10 == 0:
        print(loss)
    loss_arr.append(loss.data.numpy()[0])

param_list = list(model.parameters())
print(param_list[0].data,param_list[1].data)
```

Linear Regression

linear regression 모델 생성

```
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.01)

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

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 10 == 0:
        print(loss)
    loss_arr.append(loss.data.numpy()[0])

param_list = list(model.parameters())
print(param_list[0].data,param_list[1].data)
```

Linear Regression

linear regression 모델 생성

loss function 및
gradient descent
optimizer 생성

```python
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.01)

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

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 10 == 0:
        print(loss)
    loss_arr.append(loss.data.numpy()[0])

param_list = list(model.parameters())
print(param_list[0].data,param_list[1].data)
```

Linear Regression

linear regression 모델 생성

loss function 및
gradient descent
optimizer 생성

<training 단계>

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

```python
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.01)

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

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 10 == 0:
        print(loss)
    loss_arr.append(loss.data.numpy()[0])

param_list = list(model.parameters())
print(param_list[0].data,param_list[1].data)
```

Linear Regression

linear regression 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

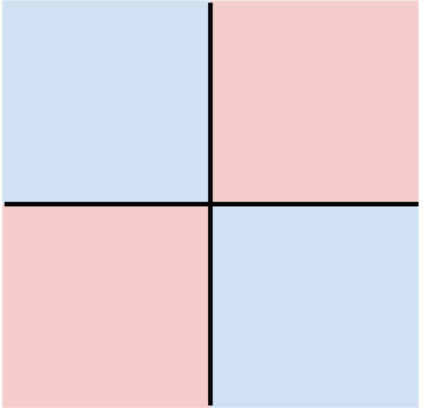
```python
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.01)

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

    loss = loss_func(output,label)
    loss.backward()
    optimizer.step()
    if i % 10 == 0:
        print(loss)
    loss_arr.append(loss.data.numpy()[0])

param_list = list(model.parameters())
print(param_list[0].data,param_list[1].data)
```

Linear Regression

Hard cases for a linear classifier

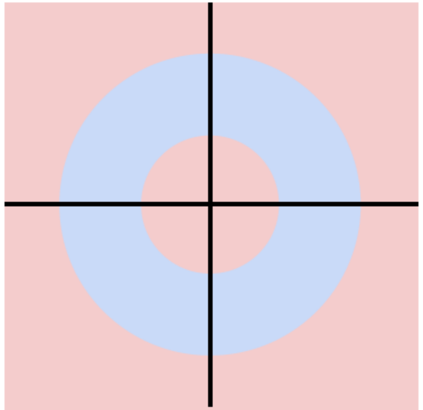
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



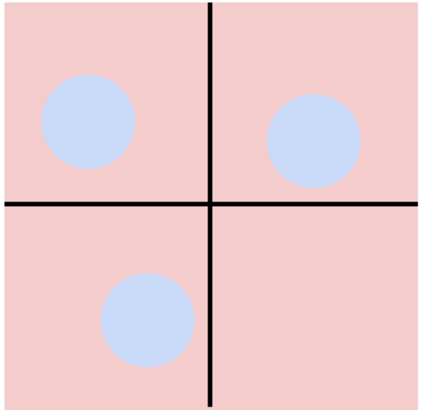
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



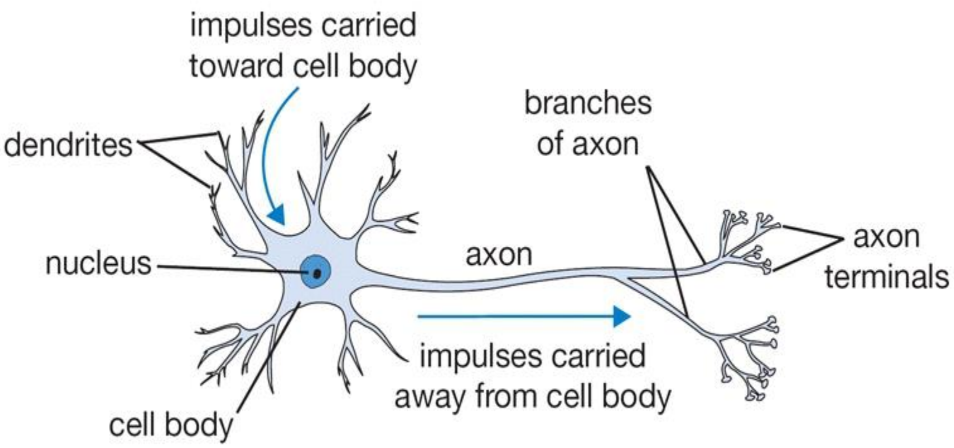
Class 1:
Three modes

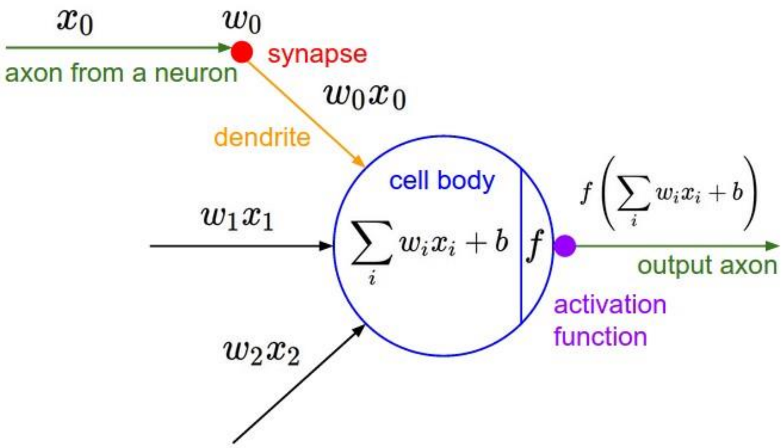
Class 2:
Everything else



(출처: CS231n_lecture2 p.59)

Neural Network



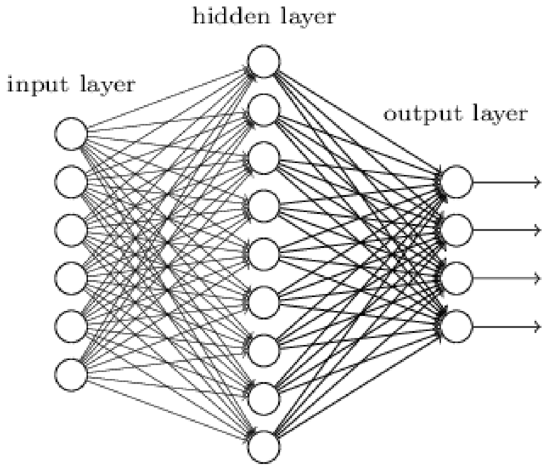


여러 자극이 들어오고 일정 기준을 넘으면 이를 다른 뉴런에 전달하는 구조

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

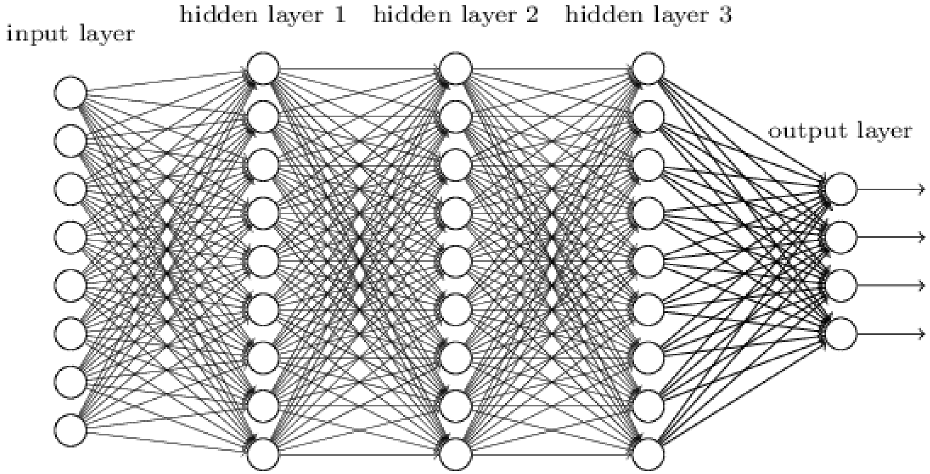
Neural Network

"Non-deep" feedforward
neural network



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

Deep neural network



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

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

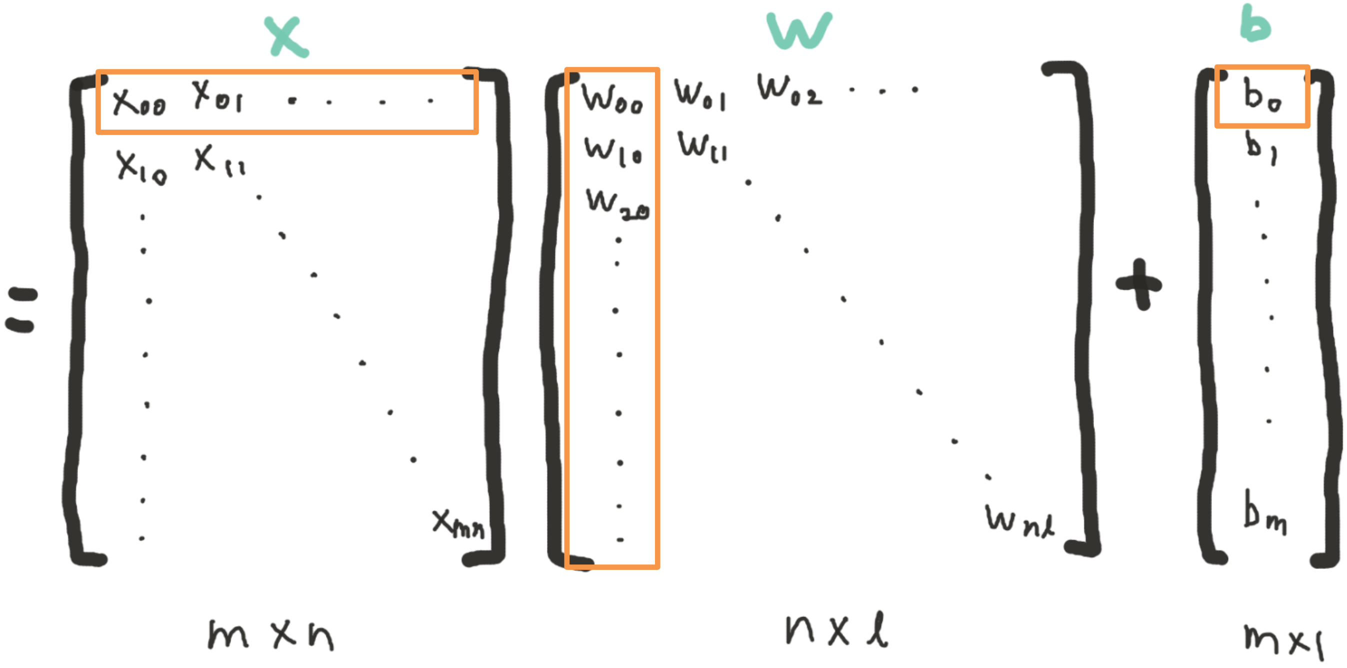
Neural Network

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

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

Neural Network

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



Neural Network

$y = act(wx+b)$

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

Neural Network

만약 activation function이 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 function이 없다면 아래의 식은 결국 linear function.

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

activation function으로 non-linearity를 추가해야 함

Neural Network

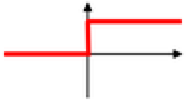
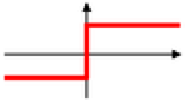
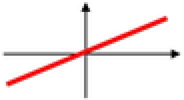
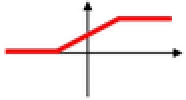
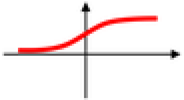
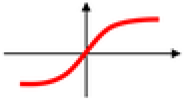
만약 activation function이 없다면 아래의 식은 결국 linear function.

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

activation function으로 non-linearity를 추가해야 함

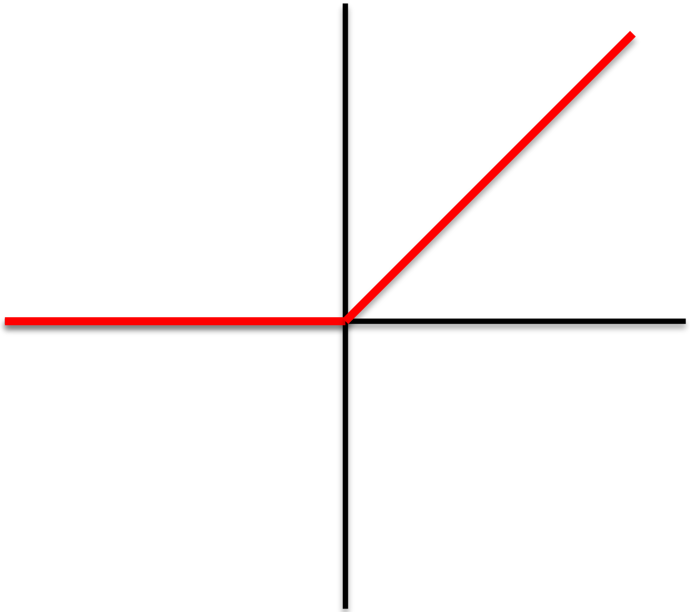
그렇다면 어떤 activation function을 써야 할까?

Neural Network

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

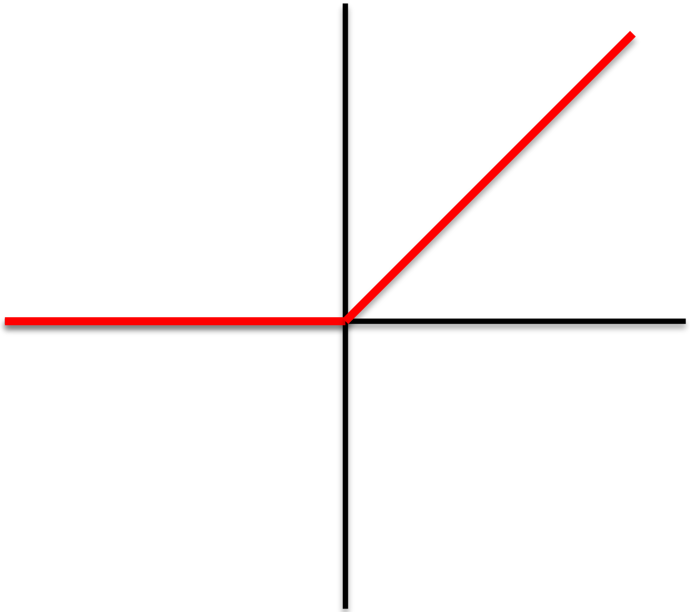
(출처: http://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html)

Neural Network



(출처: http://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html)

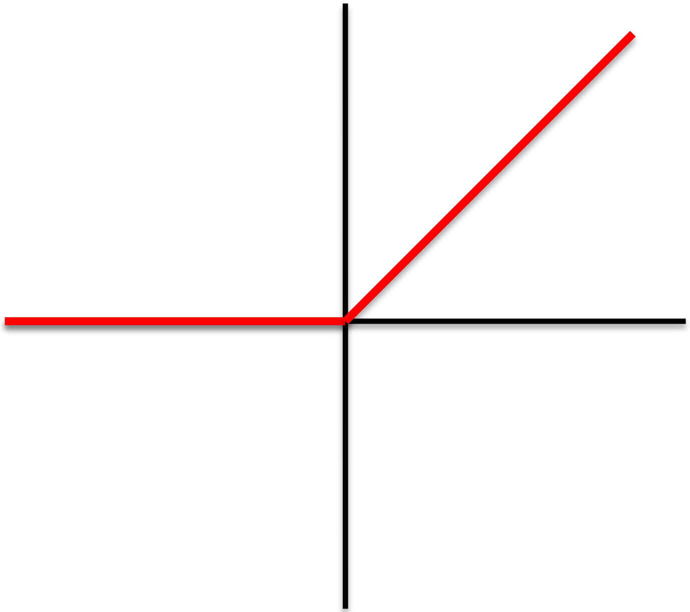
Neural Network



Rectified Linear Unit (ReLU)

(출처: http://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html)

Neural Network

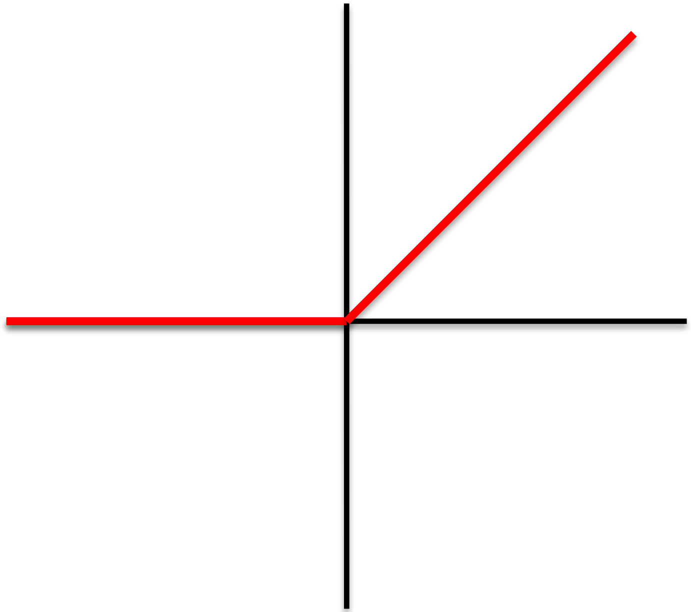


Rectified Linear Unit (ReLU)

$$f(x) = \max(0,x)$$

(출처: http://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html)

Neural Network



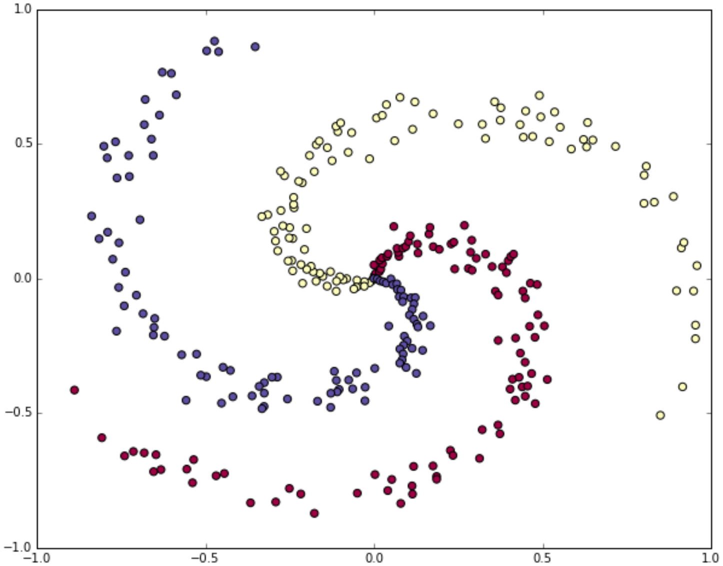
Rectified Linear Unit (ReLU)

$$f(x) = \max(0,x)$$

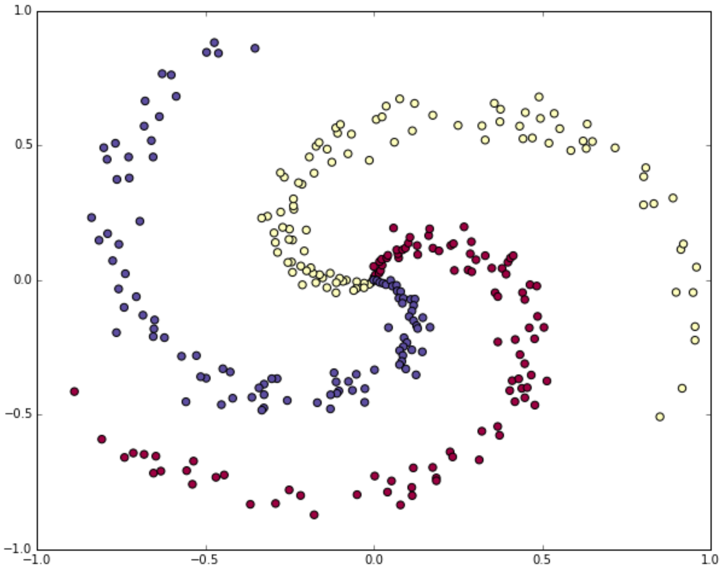
기존의 sigmoid와 tanh로는 학습이 잘 안됐었는데 relu는 gradient의 전달이 좋아서 default로 사용되고 있음

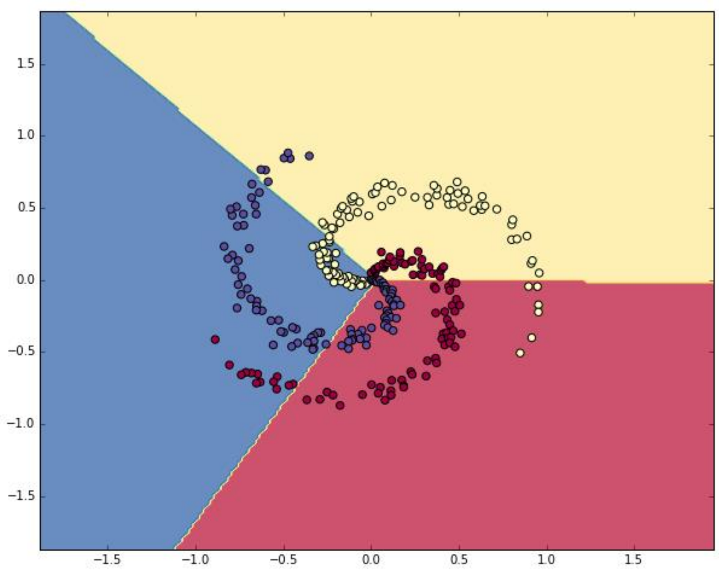
(출처: http://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html)

Neural Network

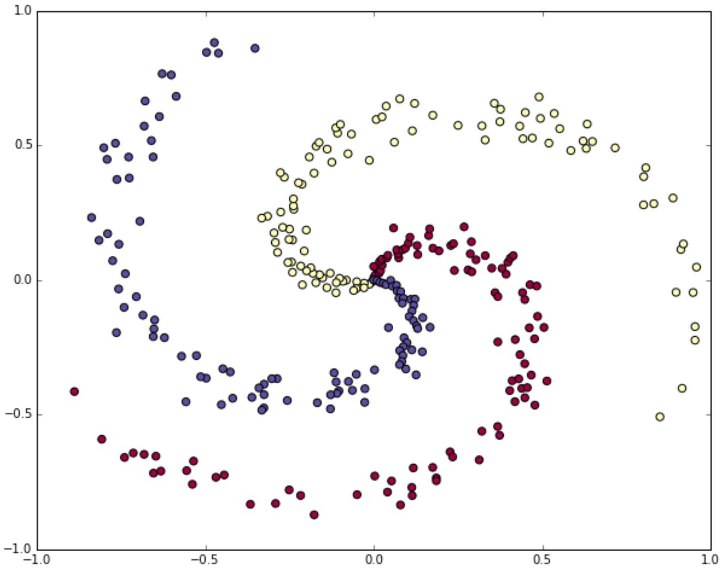


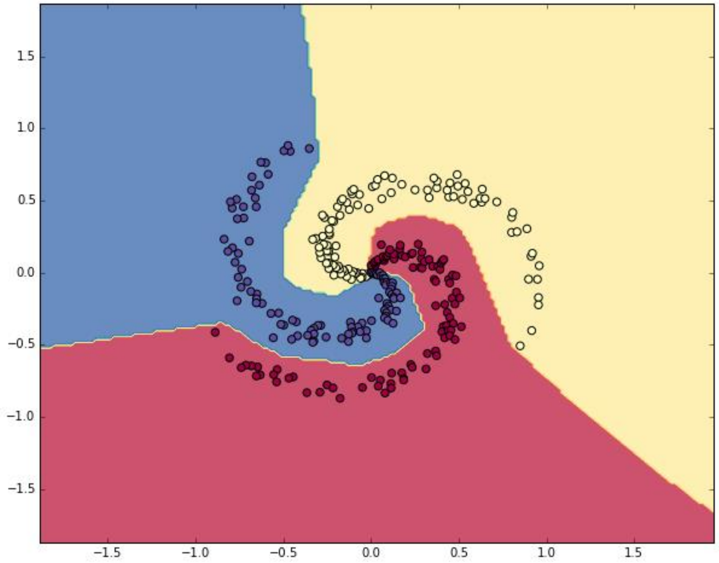
Neural Network



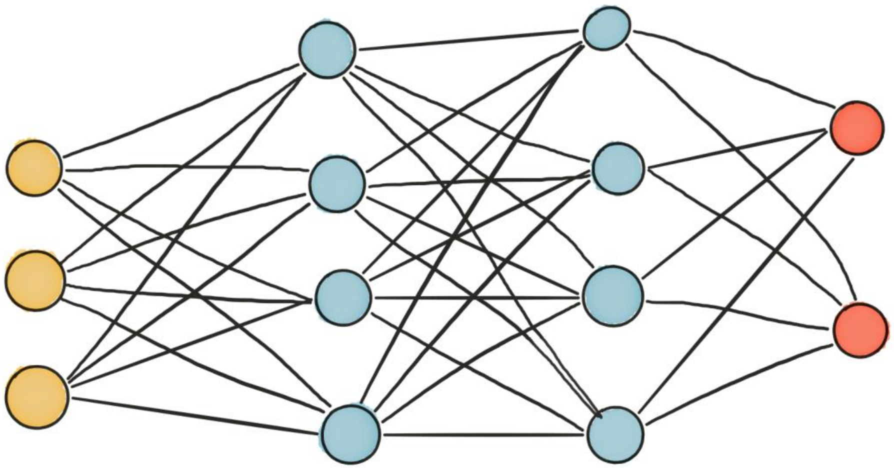


Neural Network

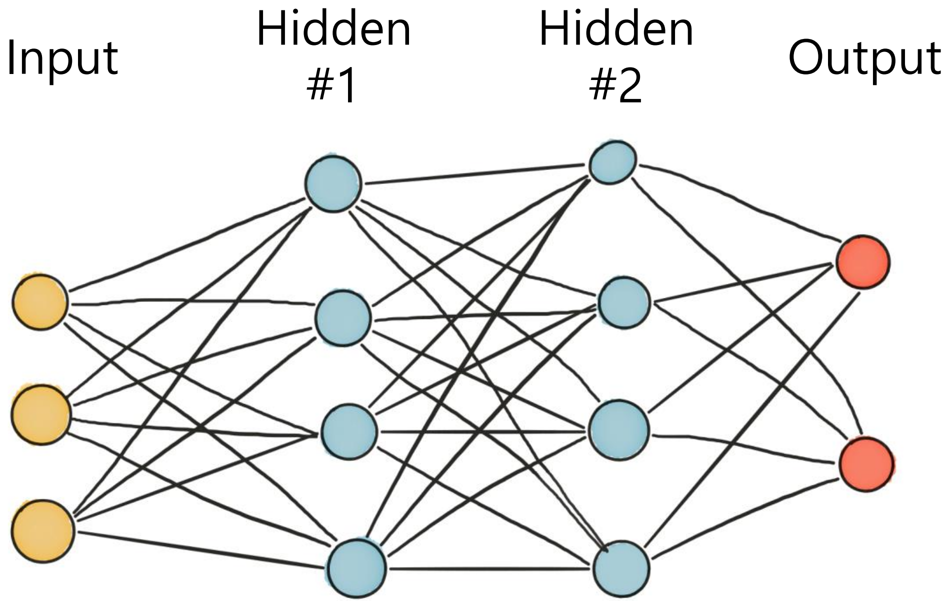




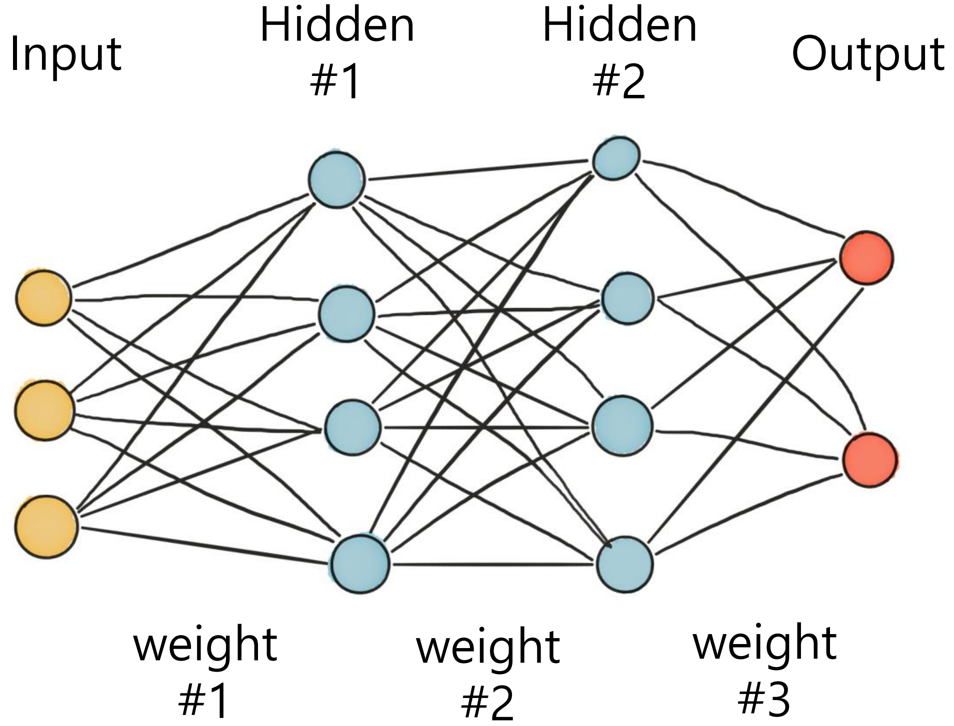
Forward & Back Prop.



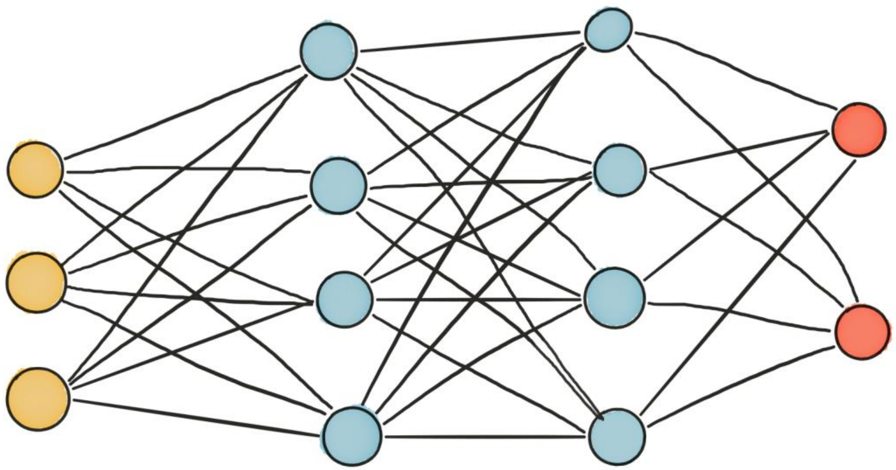
Forward & Back Prop.



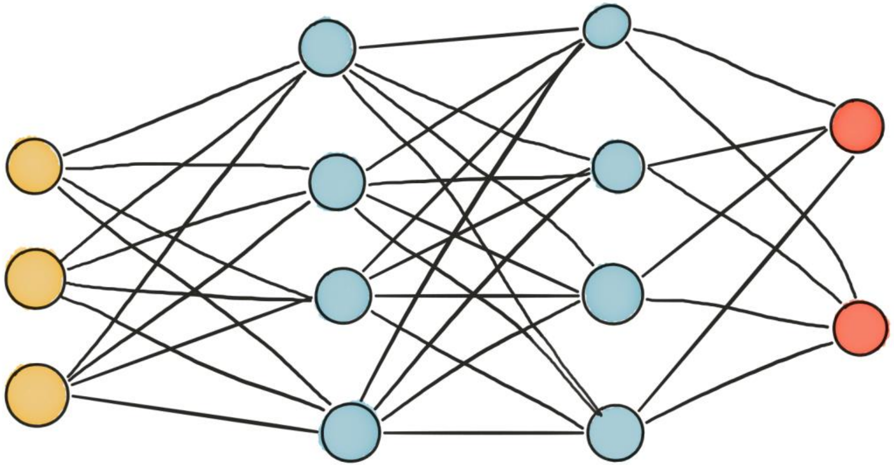
Forward & Back Prop.



Forward & Back Prop.

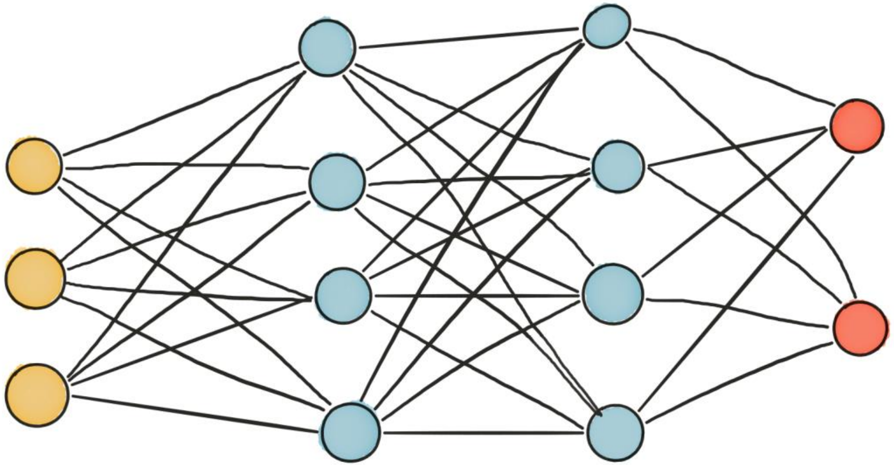


Forward & Back Prop.



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

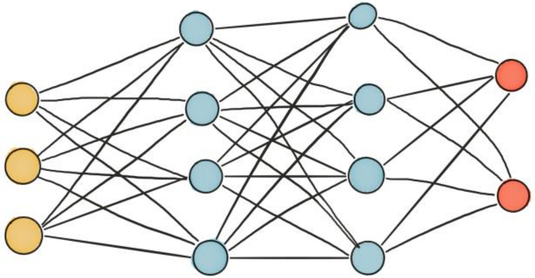
Forward & Back Prop.



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

3x4 4x4 4x2

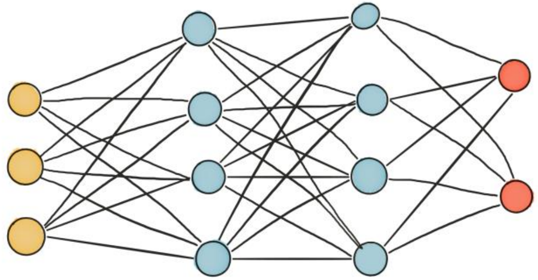
Forward & Back Prop.



쉽게 이해되도록
loss = 예측값-실제로 설정

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

Forward & Back Prop.

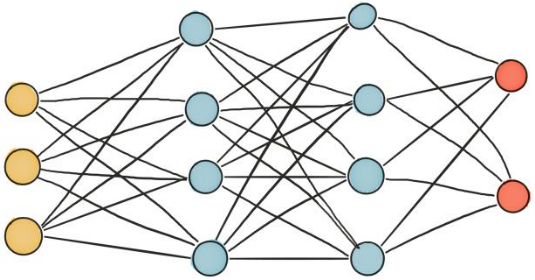


쉽게 이해되도록
loss = 예측값-실제로 설정

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

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

Forward & Back Prop.



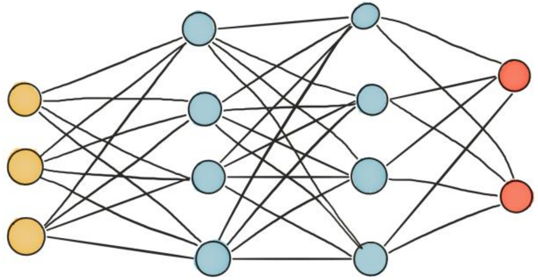
쉽게 이해되도록
loss = 예측값-실제로 설정

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

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

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

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값-실제로 설정

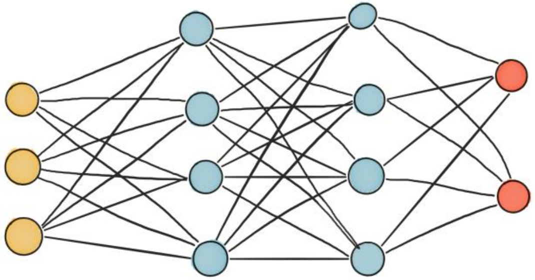
$$y^* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

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

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

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

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값-실제로 설정

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

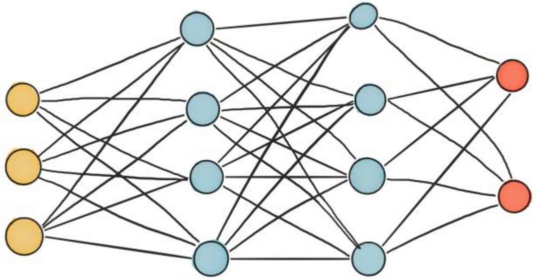
$$\begin{aligned} loss &= y^* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

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

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

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

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값-실제로 설정

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

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

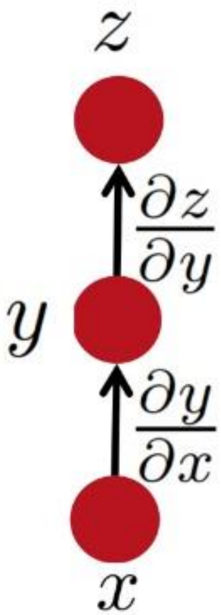
$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

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

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

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

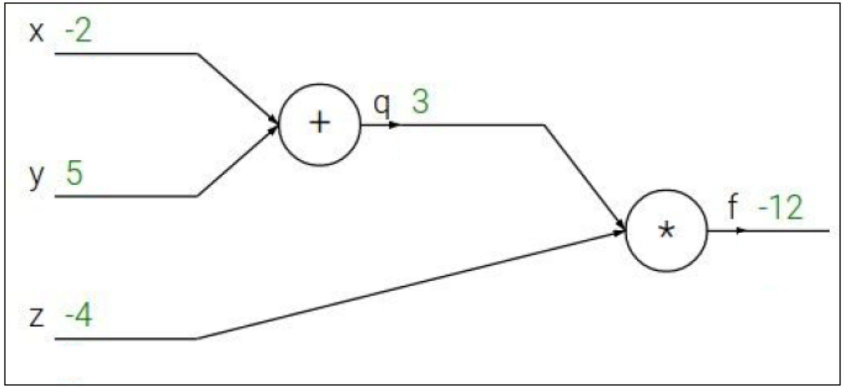
$$f(x, y, z) = (x + y)z$$

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

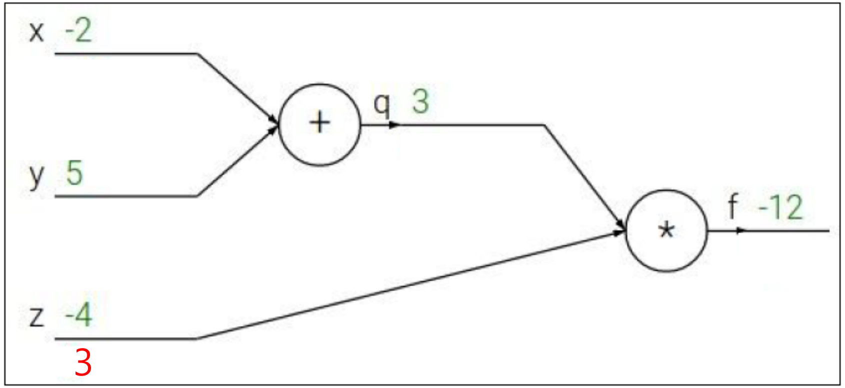
$$f(x,y,z) = (x+y)z$$

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

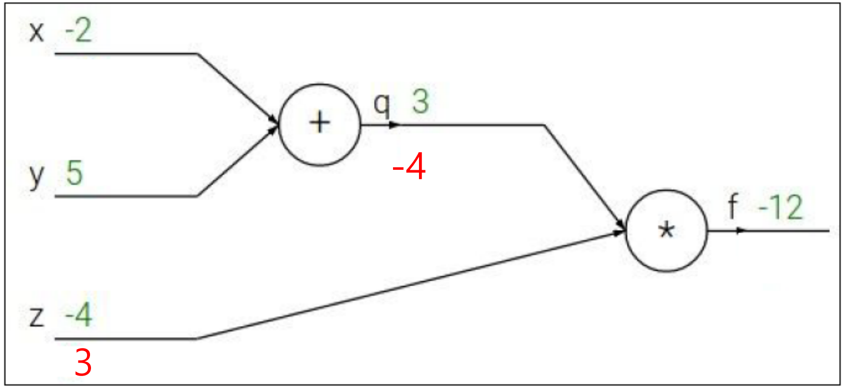
$$f(x, y, z) = (x + y)z$$

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

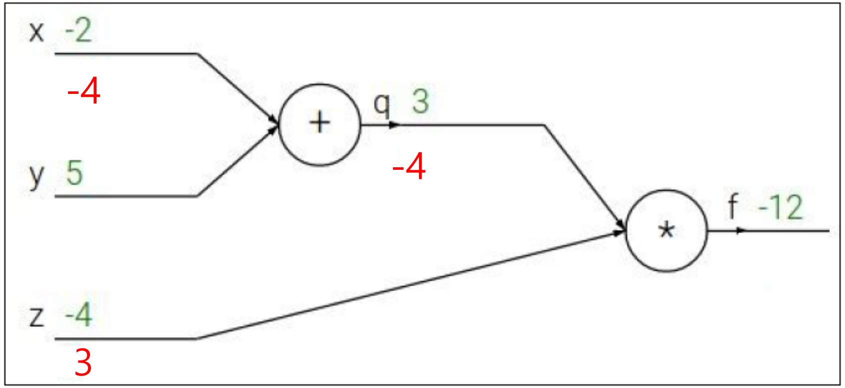
$$f(x,y,z) = (x+y)z$$

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



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

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

(출처: cs231n_lecture4 p.13)

Forward & Back Prop.

Backpropagation: a simple example

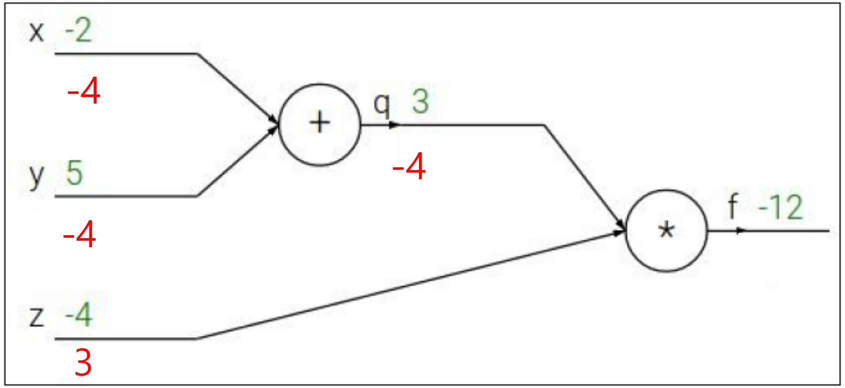
$$f(x,y,z) = (x+y)z$$

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

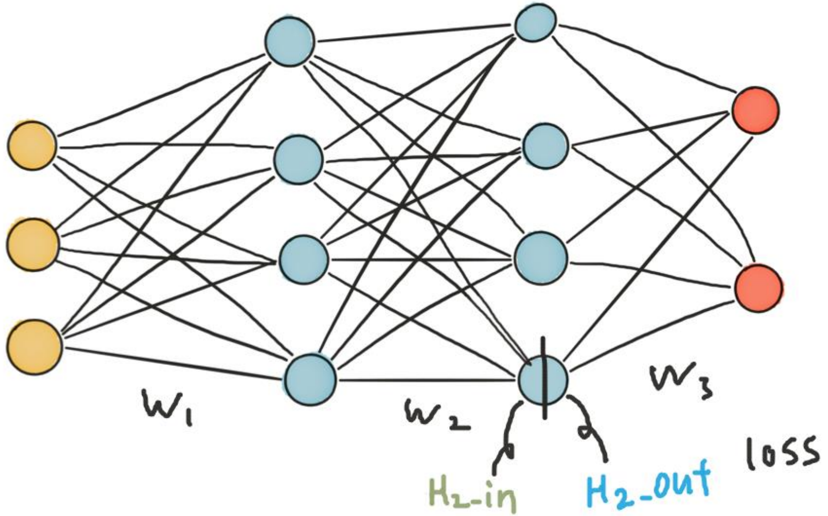


$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

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

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

Forward & Back Prop.



$$loss = W_3 \times \underbrace{sig(\underbrace{w_2 \times sig(w_1 x + b_1) + b_2}_{H_2\_in})}_{H_2\_out} + b_3 - y$$

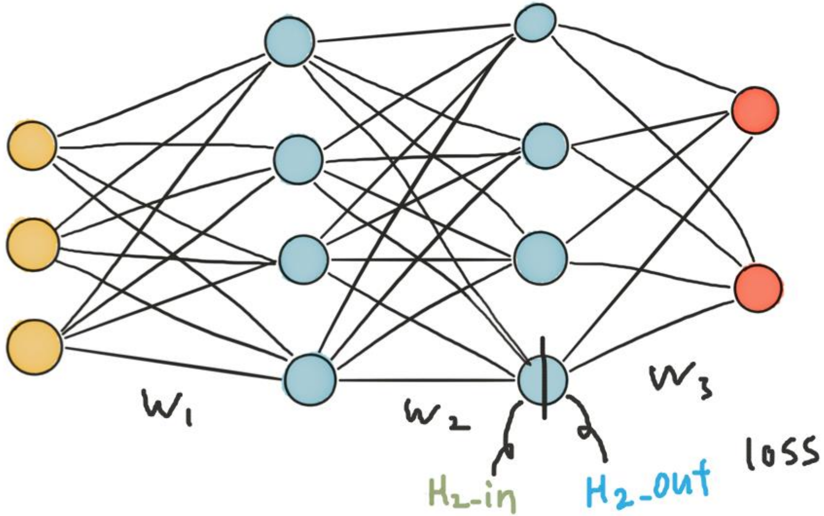
$$\frac{\partial loss}{\partial W_2} = \frac{\partial loss}{\partial H_2\_out} \times \frac{\partial H_2\_out}{\partial H_2\_in} \times \frac{\partial H_2\_in}{\partial W_2}$$

loss 중 H_2_out의 비중

H_2_out 중 H_2_in의 비중

H_2_in 중 W_2의 비중

Forward & Back Prop.



$$loss = W_3 \times \underbrace{sig(\underbrace{w2 \times sig(w_1 x + b_1) + b_2}_{H_2\_in})}_{H_2\_out} + b_3 - y$$

$$\frac{\partial loss}{\partial W_2} = \frac{\partial\, loss}{\partial H_2\_out} \times \frac{\partial H_2\_out}{\partial H_2\_in} \times \frac{\partial H_2\_in}{\partial W_2}$$

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

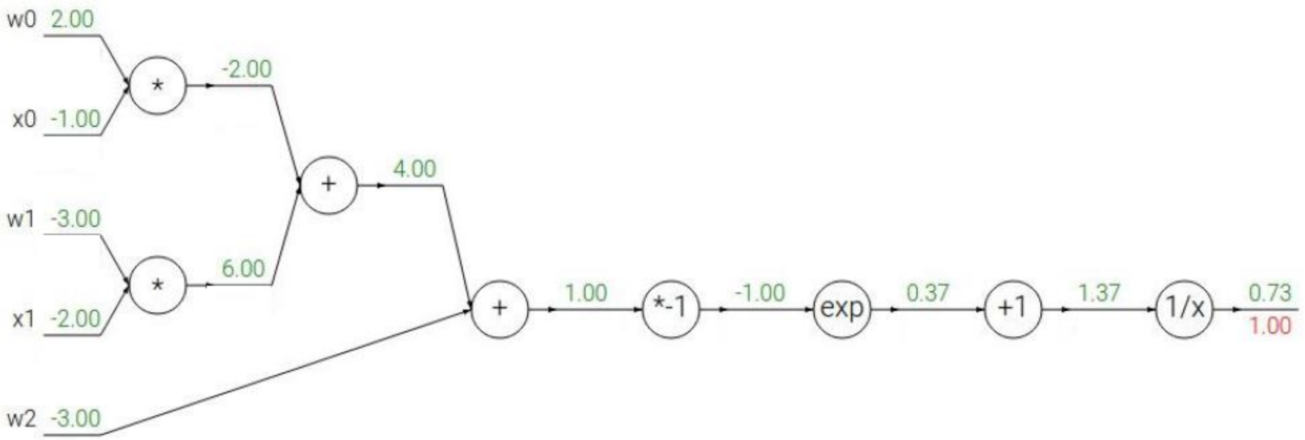
$$\sigma(x)' = \frac{\delta\{1+e^{-x}\}^{-1}}{\delta x} = -(1+e^{-x})^{-2}-e^{-x} = \frac{e^{-x}}{(1+e^{-x})^2}$$

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

(출처: http://newsight.tistory.com/68)

Forward & Back Prop.

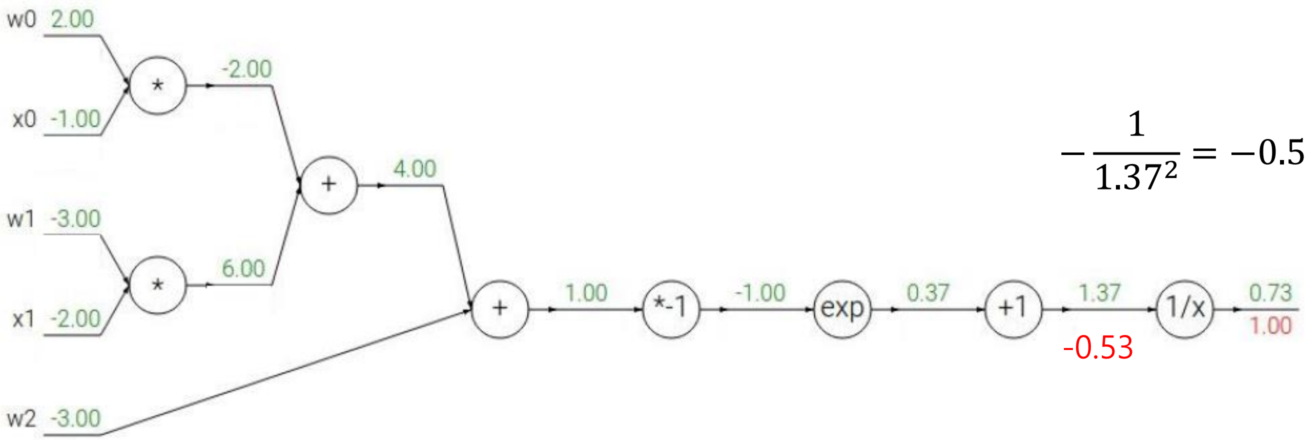
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



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

Forward & Back Prop.

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

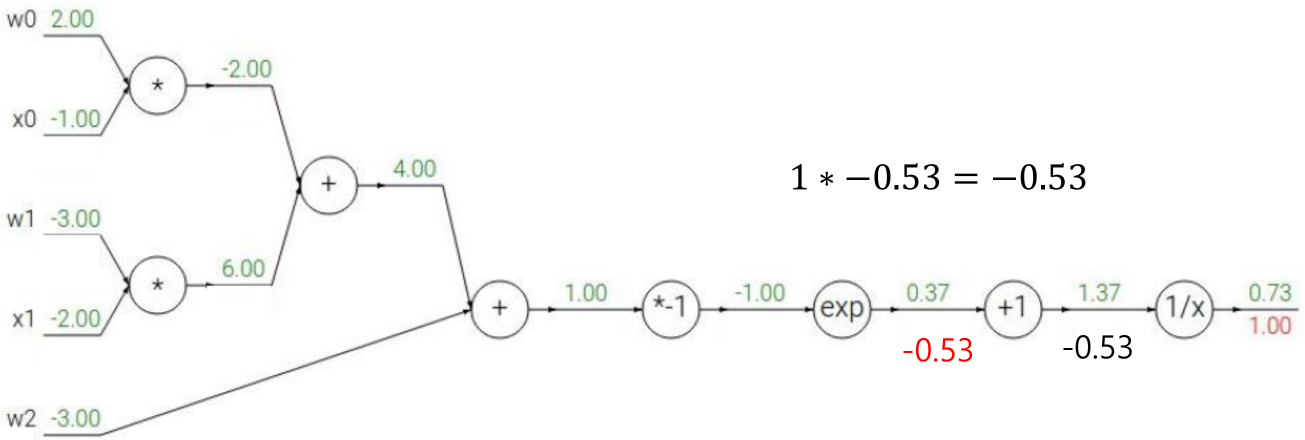


$$-\frac{1}{1.37^2} = -0.53$$

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

Forward & Back Prop.

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



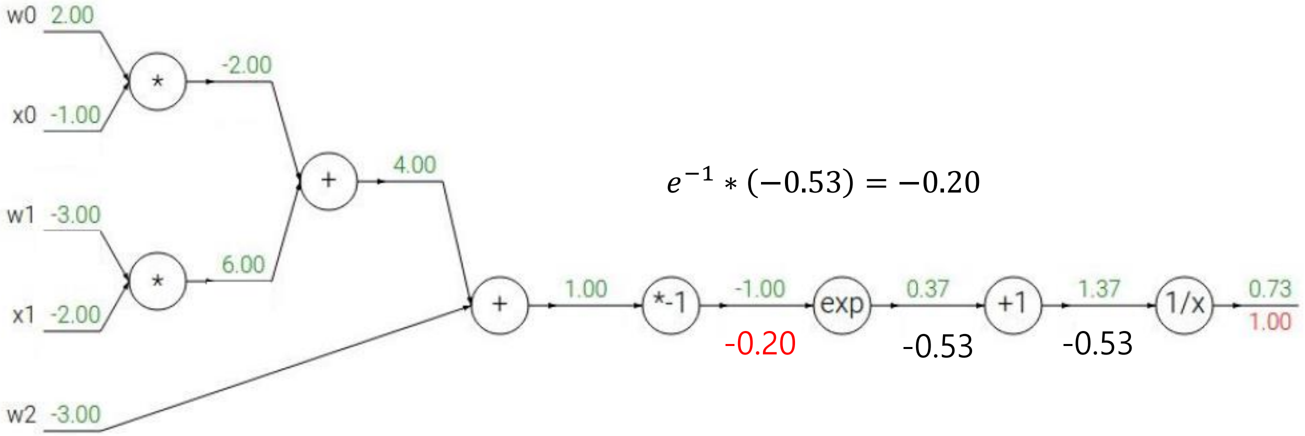
$1 * -0.53 = -0.53$

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

(출처: cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

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

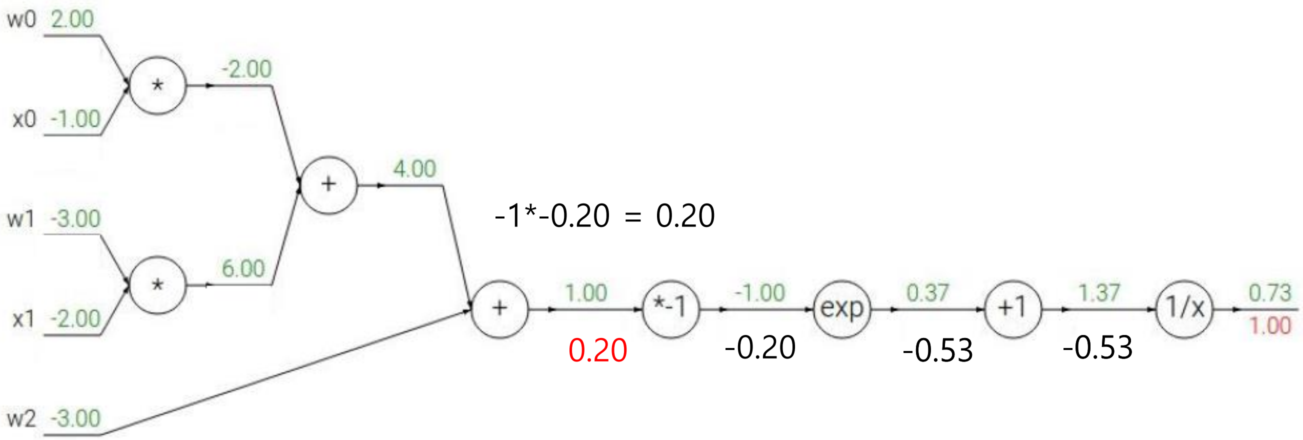
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

(출처: cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

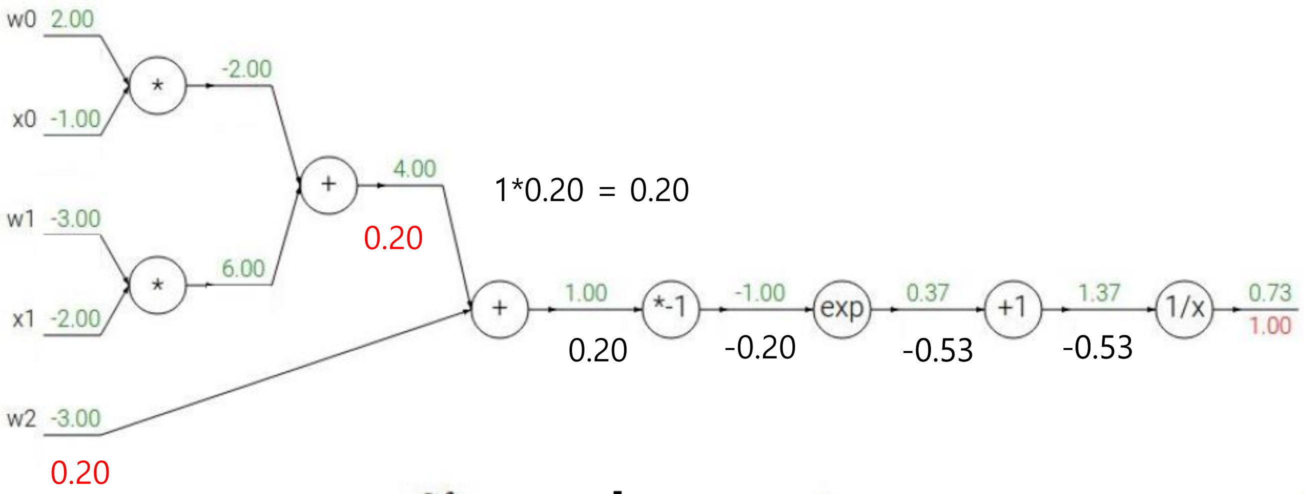
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

(출처: cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

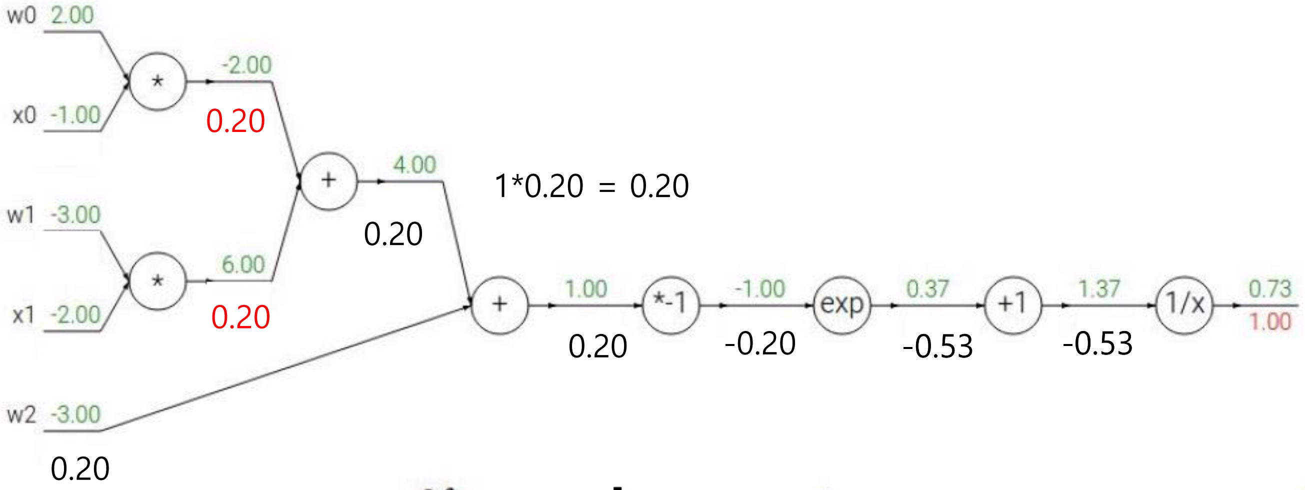
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

(출처: cs231n_lecture4 p.31)

Forward & Back Prop.

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



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

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

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

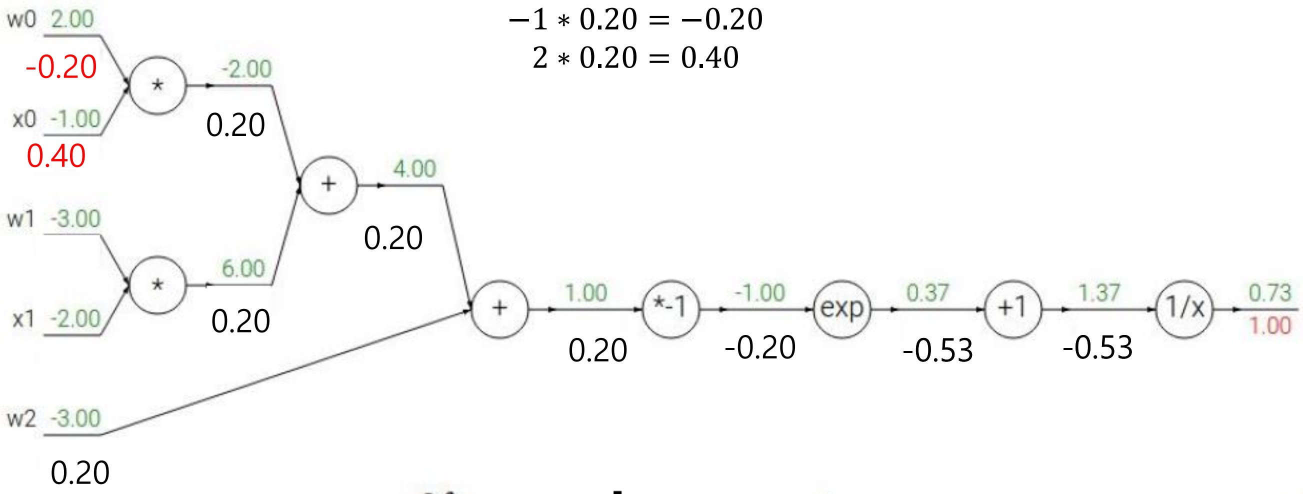
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

(출처: cs231n_lecture4 p.31)

Forward & Back Prop.

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

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



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

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

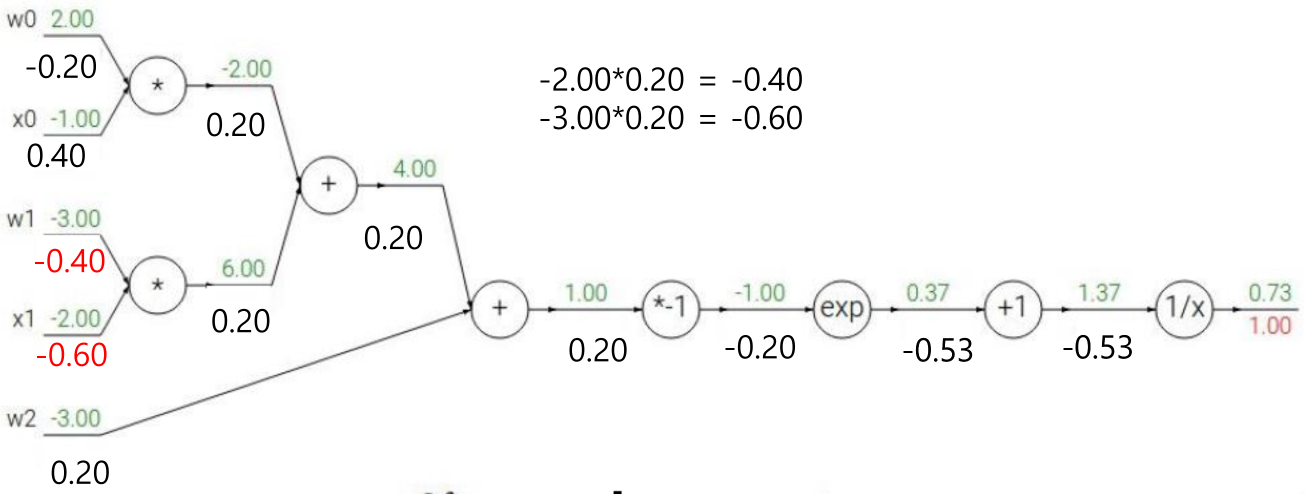
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

(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.

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

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

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

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

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

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

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

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

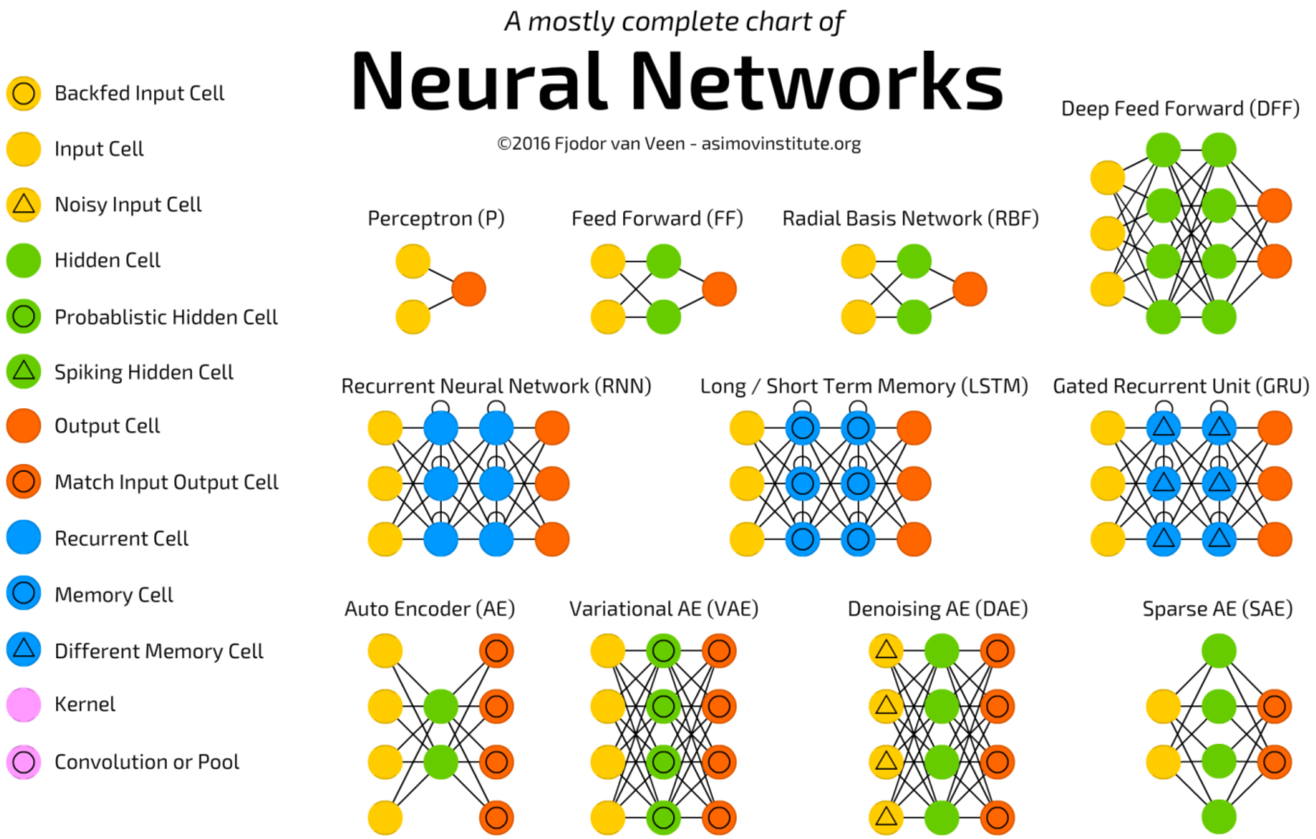
$$\frac{\partial \alpha}{\partial w_0} = -x_0 = 1$$

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

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

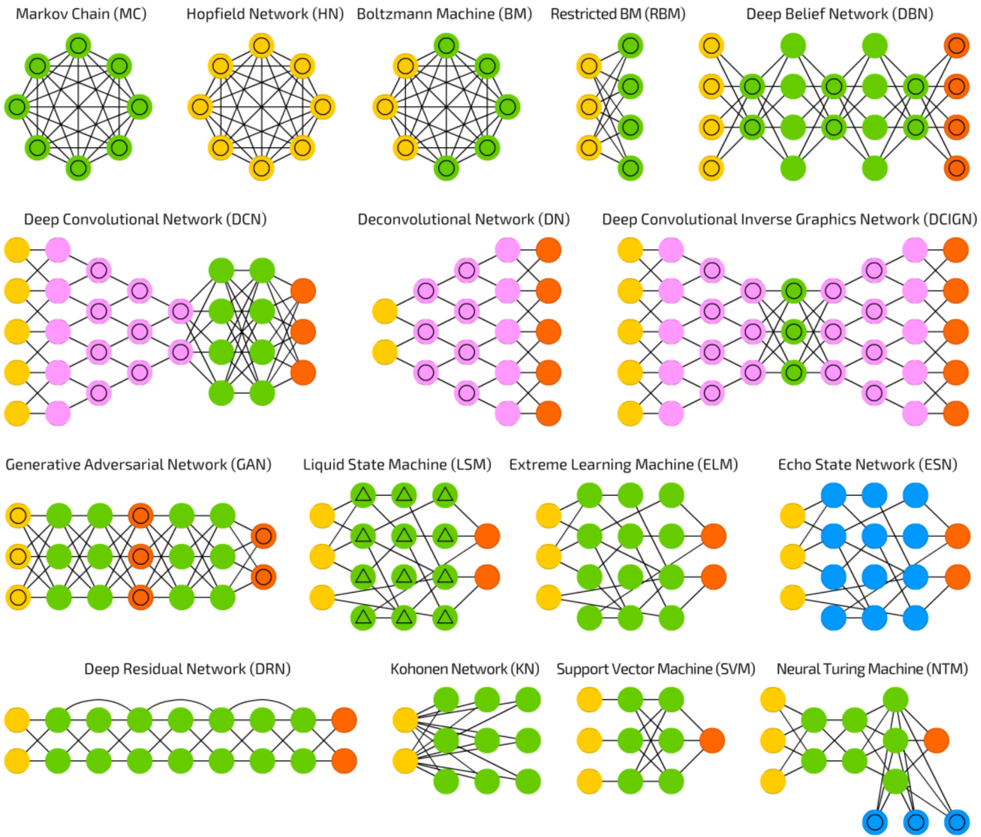
$$= -0.1959 \approx -0.2$$

Neural Network



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

Neural Network



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

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

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

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

```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

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)
```

Q&A