BE530 – Medical Deep Learning

- Artificial Neural Networks -

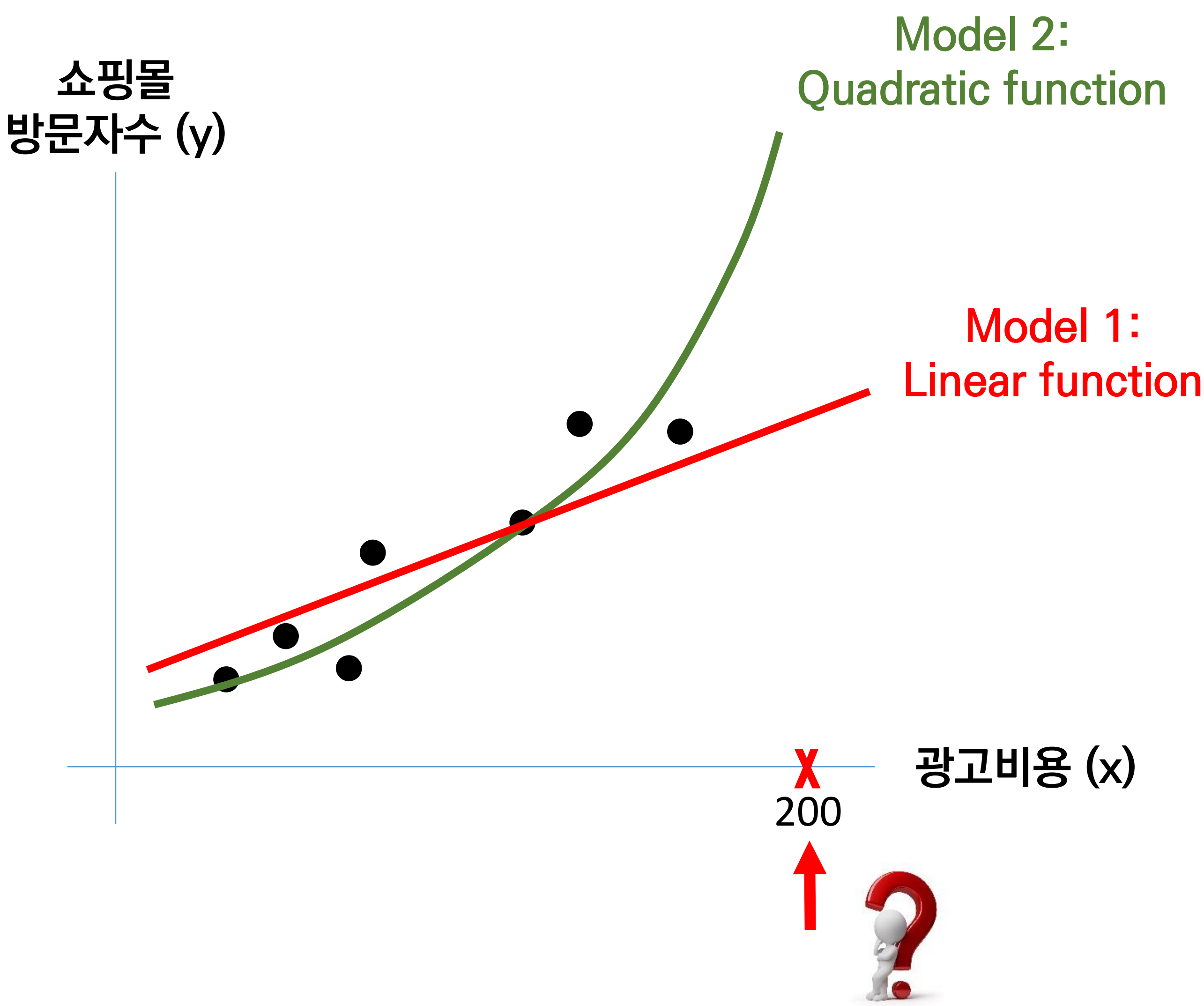
Byoung-Dai Lee

Division of AI Computer Science and Engineering

Kyonggi University

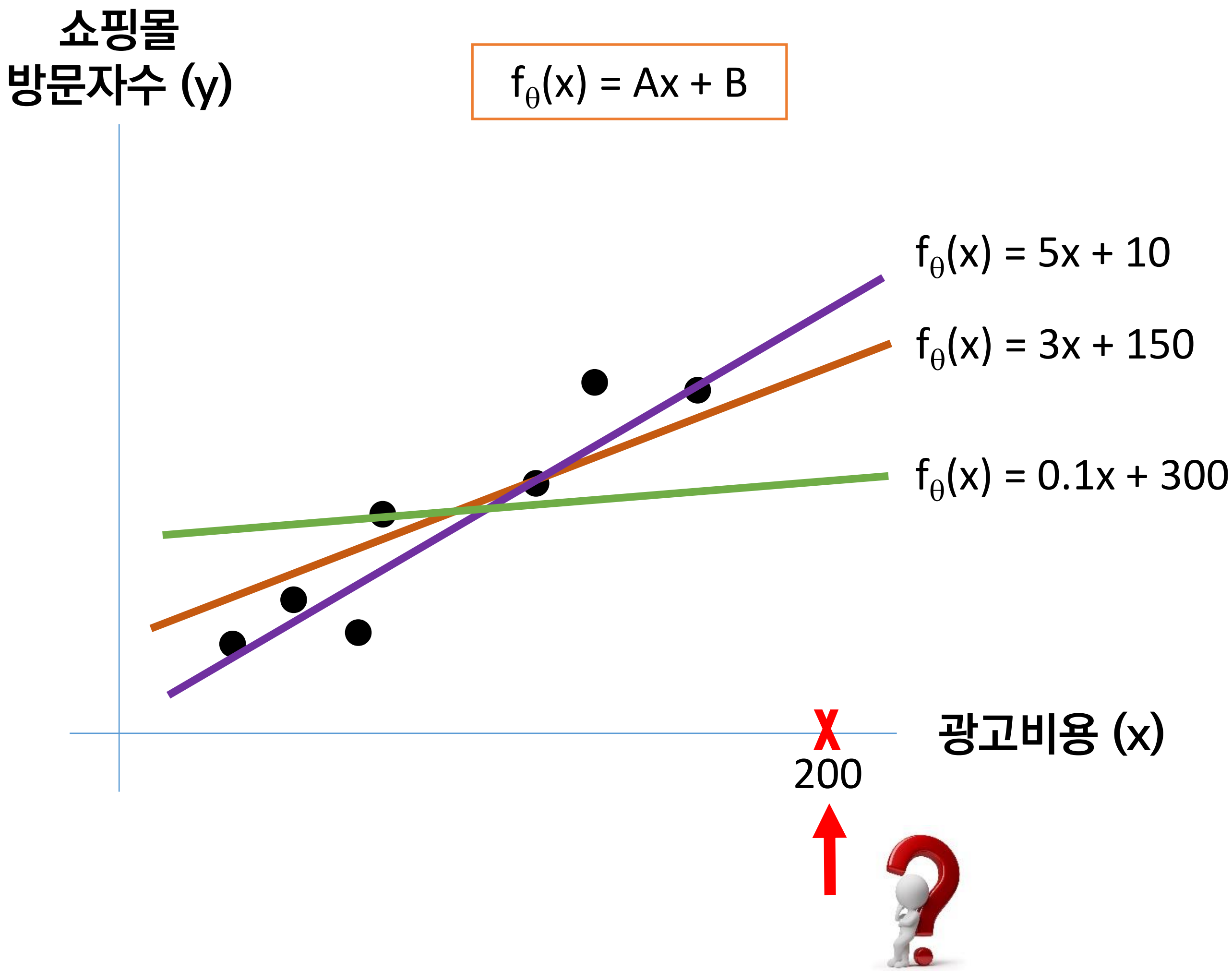
Simple Regression (1)

광고비	방문자수
58	374
70	385
81	375
84	401
95	481
107	502
113	495

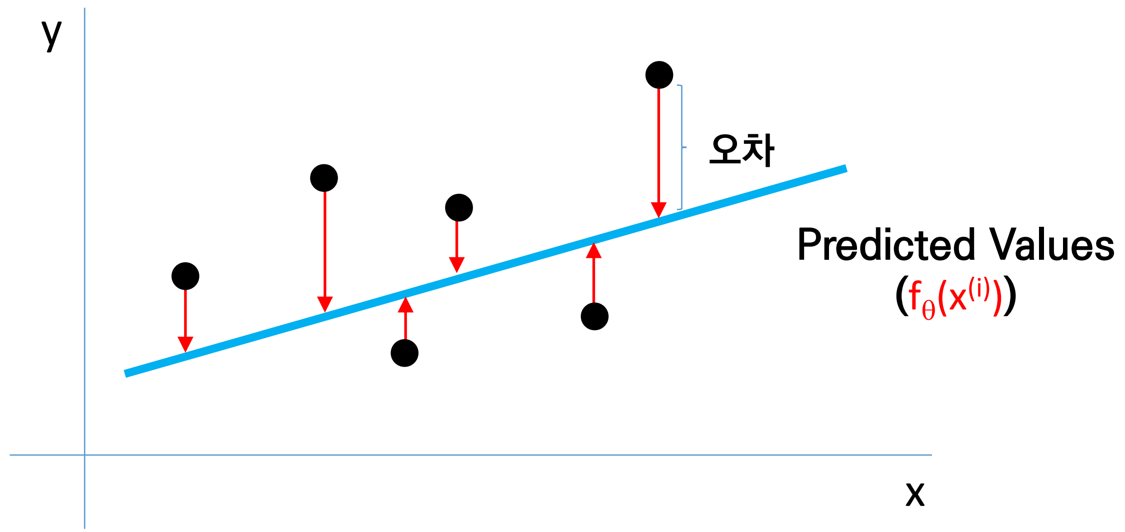


Simple Regression – (2)

광고비	방문자수
58	374
70	385
81	375
84	401
95	481
107	502
113	495



Simple Prediction – (3)



Cost Function

$$E(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f_\theta(x^{(i)})\right)^2$$

광고비	방문자수	5x+10	3x+150	0.1x+300
58	374			
70	385			
81	375			
84	401	2401	937.57	17648.46
95	481			
107	502			
113	495			

If Deep Learning is applied

① A linear function

② $f_\theta(x) = 5x + 10$

$f_\theta(x) = 3x + 150$

$f_\theta(x) = 0.1x + 300$

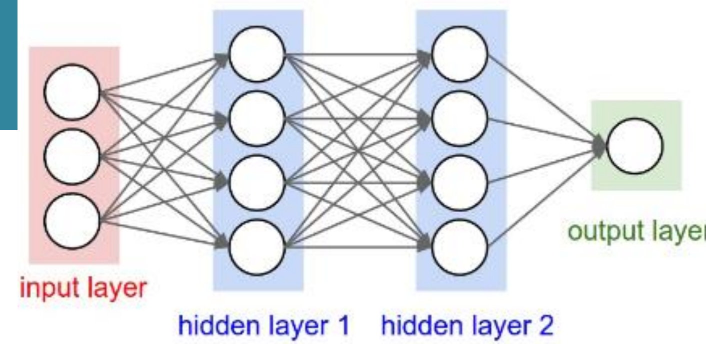
주어진 학습데이터를 이용하여
E(θ)를 최소화시키는 모델의 θ를 자동으로 찾아냄

➡ Gradient Decent Method

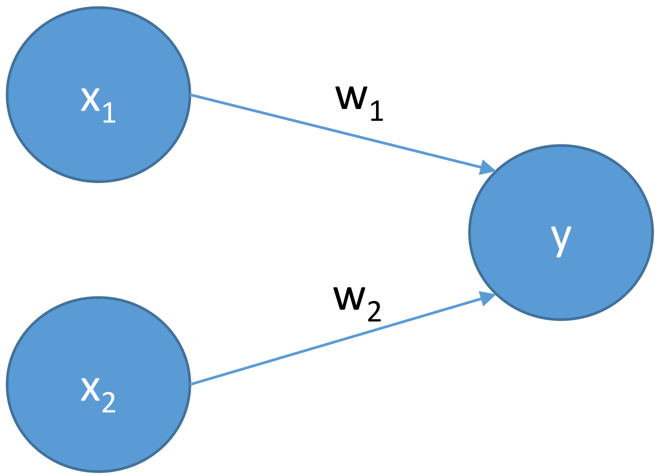
③ $$E(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f_\theta(x^{(i)})\right)^2$$

④ $\boxed{f_\theta(x) = 3x + 150}$

Perceptron (1)



- A Perceptron with 2 inputs



$$y = \begin{cases} 0 \ (x_1 w_1 + x_2 w_2 \leq \theta) \\ 1 \ (x_1 w_1 + x_2 w_2 > \theta) \end{cases}$$

※ **가중치 (weight: w_1, w_2)**

- 각 신호(e.g., x_1, x_2)가 결과에 주는 영향력을 조절하는 요소
- 가중치가 클수록 해당 신호가 그만큼 더 중요함을 의미함

Perceptron (2)

■ Weights and Bias

$$y = \begin{cases} 0 \ (x_1 w_1 + x_2 w_2 \le \vartheta) \\ 1 \ (x_1 w_1 + x_2 w_2 > \vartheta) \end{cases} \Rightarrow y = \begin{cases} 0 \ (b + x_1 w_1 + x_2 w_2 \le 0) \\ 1 \ (b + x_1 w_1 + x_2 w_2 > 0) \end{cases}$$

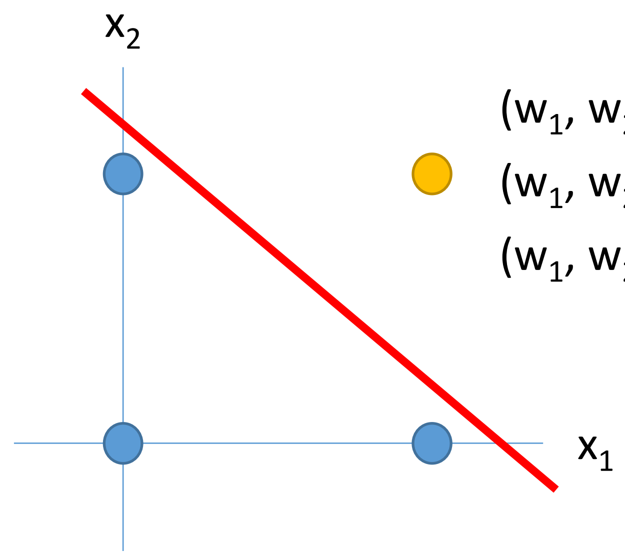
※ **가중치 (weight: w_1, w_2)**
- 각 입력 신호(e.g., x_1, x_2)가 결과에 주는 영향력을 조절하는 변수

※ **편향 (bias: b)**
- 뉴런이 얼마나 쉽게 활성화(결과로 1 출력)되는지를 조정하는 변수
- 예) b=−0.1이면 각 입력 신호와 가중치를 곱한 값들이 합이 0.1을 초과할 때만 뉴런이 활성화. b=−20.0이면 입력 신호와 가중치를 곱한 값들이 합이 20을 초과하지 않으면 뉴런이 활성화되지 않음

Perceptron (3)

AND gate

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1



$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$

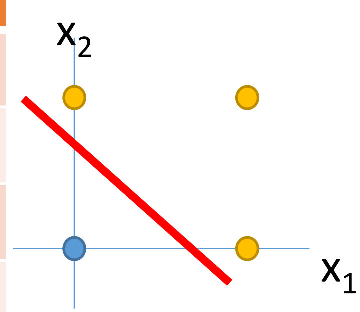
$(w_1, w_2, \theta) = (0.5, 0.5, 0.8)$

$(w_1, w_2, \theta) = (1.0, 1.0, 1.0)$

…

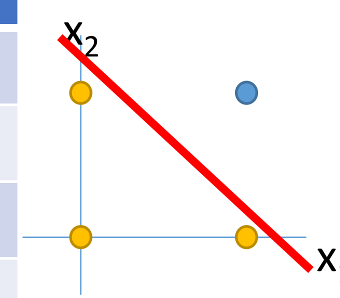
OR gate

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1



NAND gate

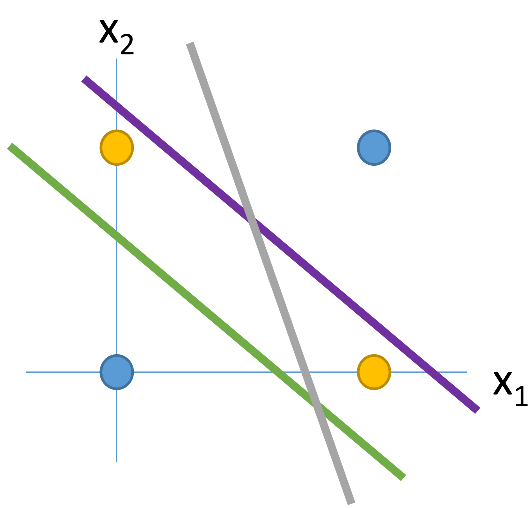
x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

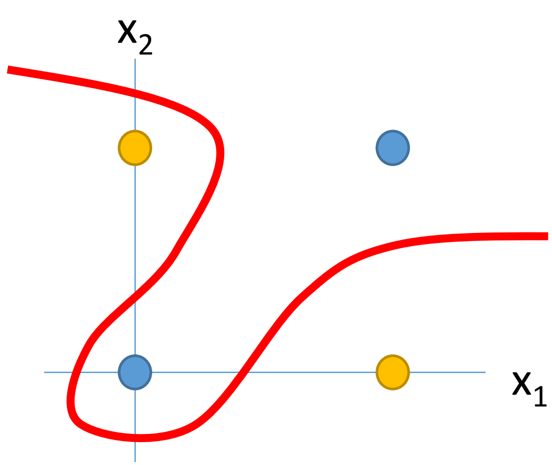


Perceptron (4)

- XOR gate
  - **Perceptron을 적용할 경우 XOR Gate에 대해 하나의 직선을 이용하여 0과 1을 구분할 수 없음**
    - Perceptron은 직선 하나로 나눈 영역에 대해서만 표현 가능

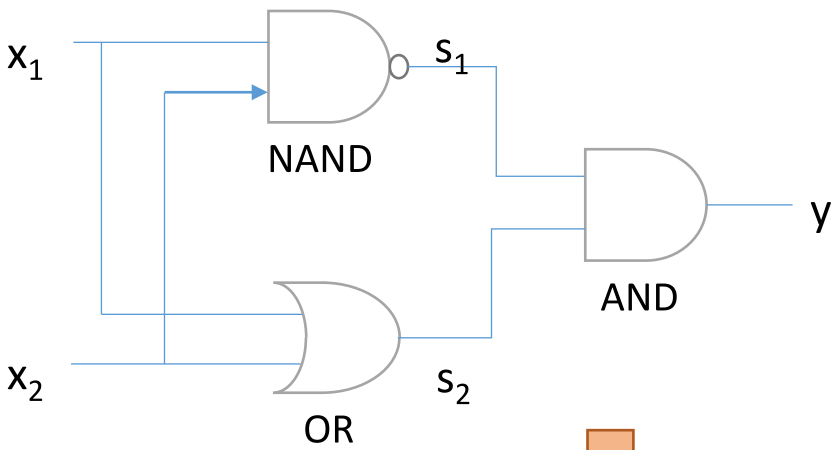
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0





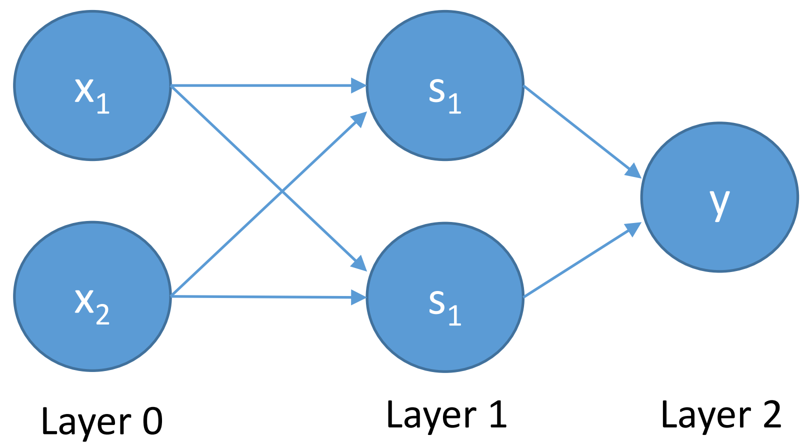
Perceptron (5)

■ Multi-layer Perceptron (MLP)



XOR Gate

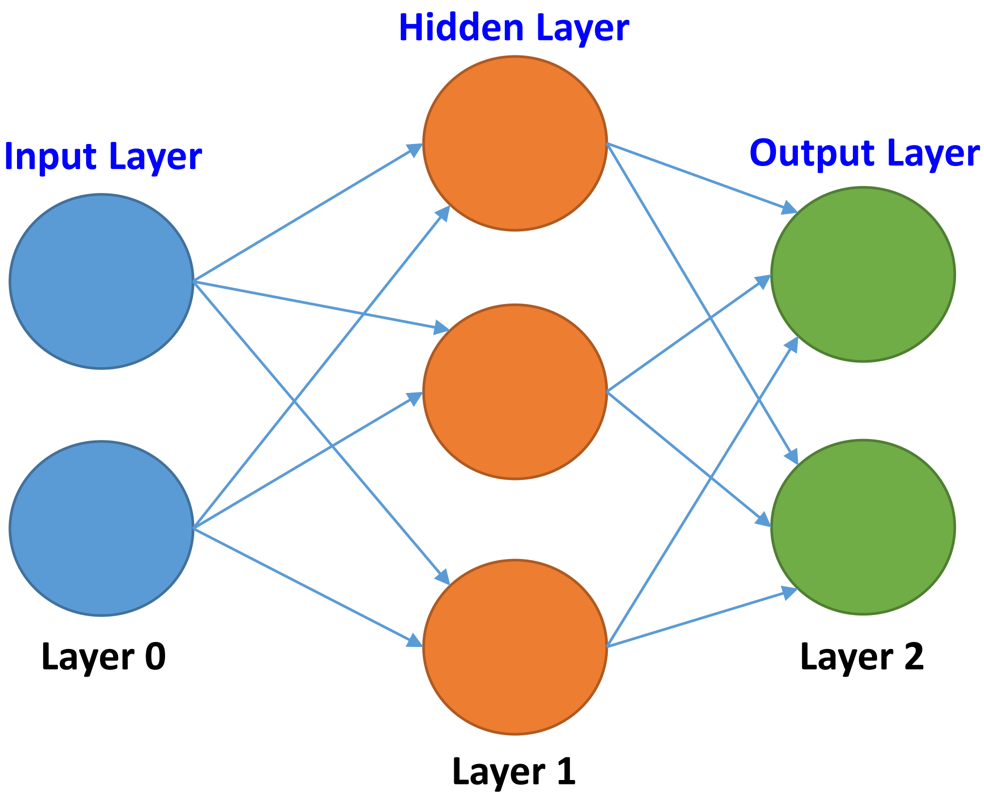
x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0



- Single Layer Perceptron ➔ 직선형 영역 표현
- Multi-Layer Perceptron ➔ 비선형 영역 표현

Artificial Neural Networks

- 2-Layer ANN

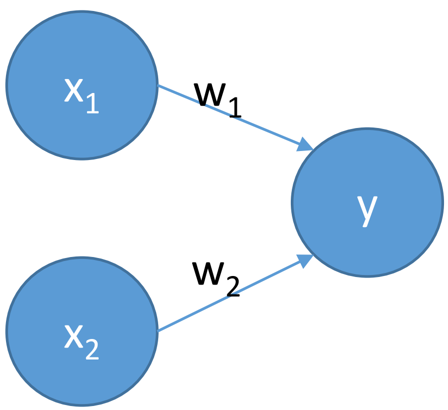


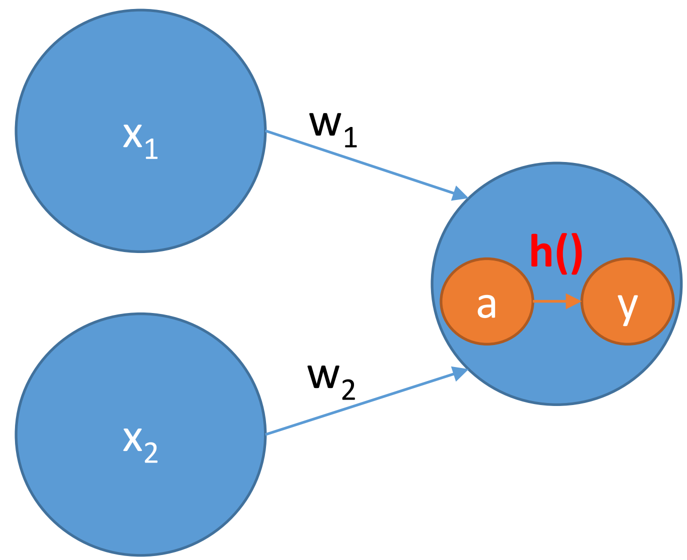
Perceptron revisited

$$y = \begin{cases} 0 \ (b + x_1w_1 + x_2w_2 \le 0) \\ 1 \ (b + x_1w_1 + x_2w_2 > 0) \end{cases}$$

$$a = b + w_1x_1 + w_2x_2$$

$$y = h(a) = \begin{cases} 0 \ (a \le 0) \\ 1 \ (a > 0) \end{cases}$$





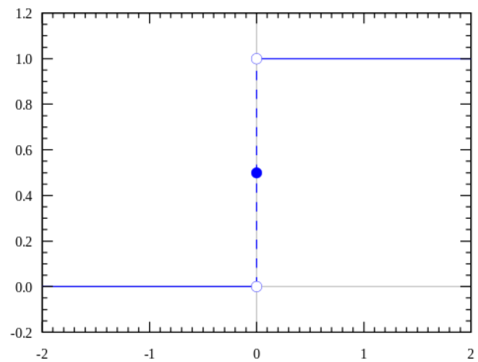
※ **h(a)**

- 활성화 함수 (Activation Function)
- 입력 신호의 총합을 출력 신호로 변환
- Ex) step, sigmoid, ReLU, …

Activation Functions

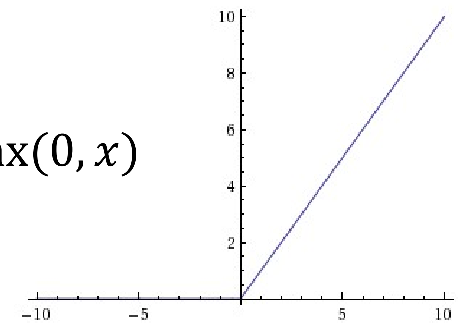
- Step function

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



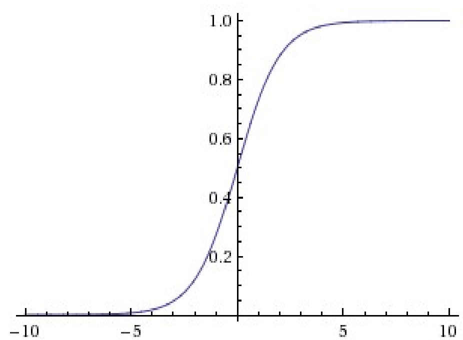
- ReLU function

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



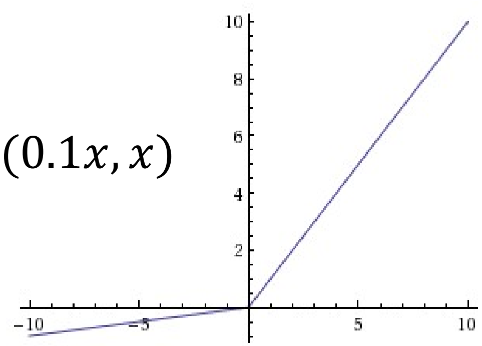
- Logistic sigmoid function

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



- Leaky ReLU function

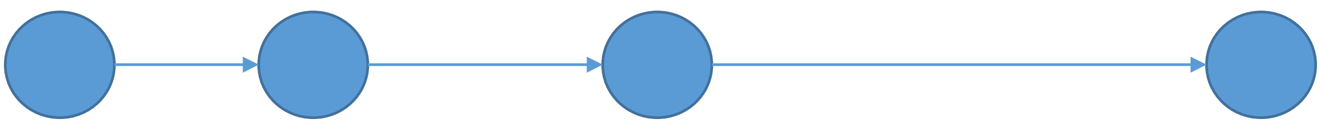
$$f(x) = \max(0.1x, x)$$



Perceptron vs. ANN

- **Perceptron과 Neural Network과의 주된 차이는 활성화 함수**
  - Perceptron ➔ step function
  - Neural networks ➔ sigmoid, ReLU, …

- **신경망에서의 활성화 함수는 비선형 함수 사용**
  - 선형 함수 사용 시 → 은닉층(Hidden Layer)을 사용하는 이점이 사라짐



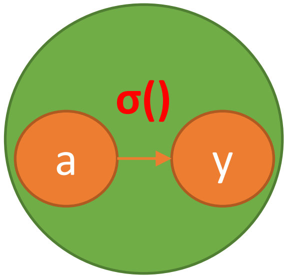
$h(a_1)=h(w_1*x)$
$=c*x$

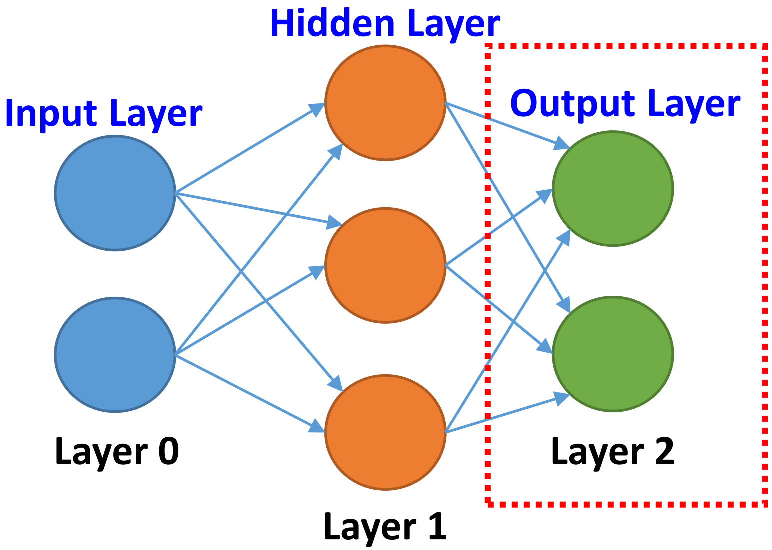
$h(a_2)=h(w_2*h(a_1))$

$y=h(a_3)=h(w_3*h(w_2*h(a_1)))=k*x$

Softmax Function

- Output Layer
  - Identity function(σ())
    - – 입력 신호 = 출력 신호





- Softmax function

$$p(c_k = 1|\boldsymbol{x}) = \frac{\exp(y_k)}{\sum_{j=1}^{C} \exp(y_j)}$$

  - 0에서 1.0사이의 출력
  - 출력의 총합 = 1.0

→ Softmax function의 출력을 『**확률**』로 해석 가능

Loss Function

■ 학습 목표

- Loss Function의 결과값을 최소화하는 가중치 매개변수 (*w*, *b*)를 찾는 것

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

N: number of training examples
$\mathcal{R}$: regularizer
$\boldsymbol{w}$: set of all parameters

■ 대표적인 손실 함수

- 평균제곱오차(MSE: Mean Squared Error) – Regression 문제

$$l = \frac{1}{2}\sum_{k}(y_k - t_k)^2 \text{ , } y_k: prediction, t_k: ground\ truth$$

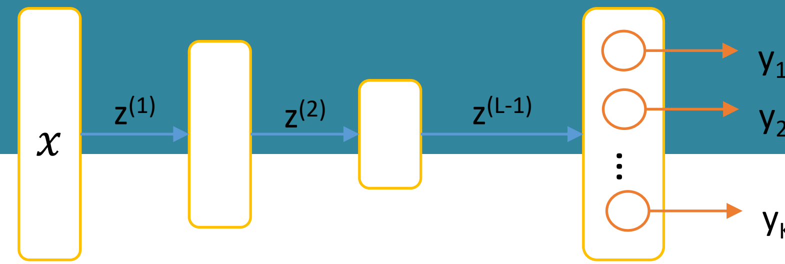
- 교차 엔트로피 오차(CEE: Cross Entropy Error) – Classification 문제

$$l = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} log y_k(x_n; w)$$

$$d_n = [d_{n1}, d_{n2}, \ldots, d_{nk}] \text{ /* one hot encoding*/}$$

KYONGGI UNIVERSITY

Cross Entropy Loss (1)



$$p(C_k|x) = y_k = z_k^{(L)} = \frac{\exp(u_k^{(L)})1}{\sum_{j=1}^{K} \exp(u_j^{(L)})}$$

$$d_n = [d_{n1}, d_{n2}, \ldots, d_{nK}] \text{ /* GT */}$$

$$p(d|x) = \prod_{k=1}^{K} p(C_k|x)^{d_k}$$

L(w) : 최대우도법 (maximum likelihood estimation)에 따른 w의 우도

$$L(w) = \prod_{n=1}^{N} p(d_n|x_n; w) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|x)^{d_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} (y_k(x; w))^{d_{nk}}$$

Objective: *L(w)*를 최대화하는 w를 구할 수 있도록 학습

*L(w)*에 로그를 취하고 부호 반전

Objective: *E(w)*를 최소화하는 w를 구할 수 있도록 학습

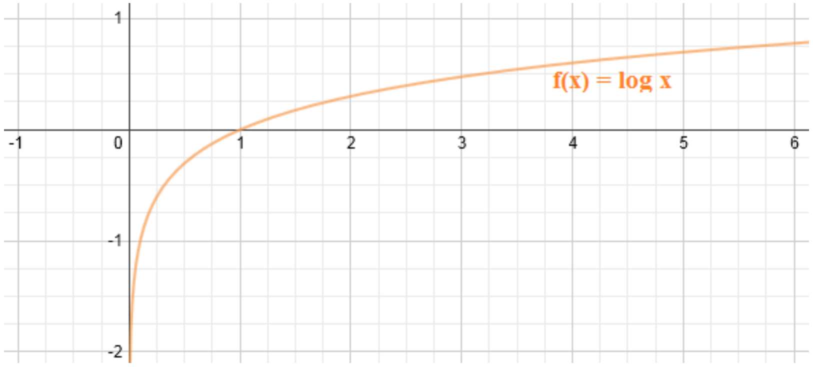
$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk}\log(y_k(x_n; w))$$

Cross Entropy Loss (2)

- Cross entropy loss with Softmax function

$$l(\boldsymbol{w}, \boldsymbol{x}, t) = -\sum_{k=1}^{C} t^{(k)} \log \underbrace{p(c_k|\boldsymbol{x})}_{\text{The output of the Softmax function}}$$

$$p(c_k = 1|\boldsymbol{x}) = \frac{\exp(y_k)}{\sum_{j=1}^{C} \exp(y_j)}$$



Gradient Descent Method (1)

- Given E(w), the gradient descent is defined as

$$\nabla E \equiv \frac{\partial E}{\partial w} = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_M}\right], M: w\text{의 성분 수}$$

- Gradient Descent Method
  - 현재의 w를 음의 기울기 방향($-\nabla E$)로 조금씩 움직이는 것을 여러 번 반복
  - 현재의 가중치를 $w^{(t)}$, 움직인 후의 가중치 $w^{(t+1)}$ 를 라고 할 때 다음과 같이 갱신

$$w^{(t+1)} = w^{(t)} - \epsilon \nabla E, \epsilon: learning\ rate$$

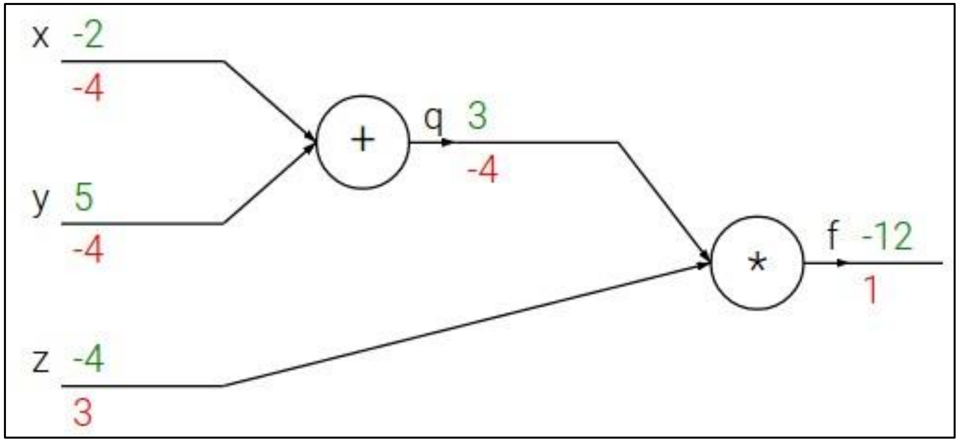
– 학습률(ϵ)을 결정하는 것은 학습에서 매우 중요한 요소

Gradient Descent Method (2)

Backpropagation (1)

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

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



* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (2)

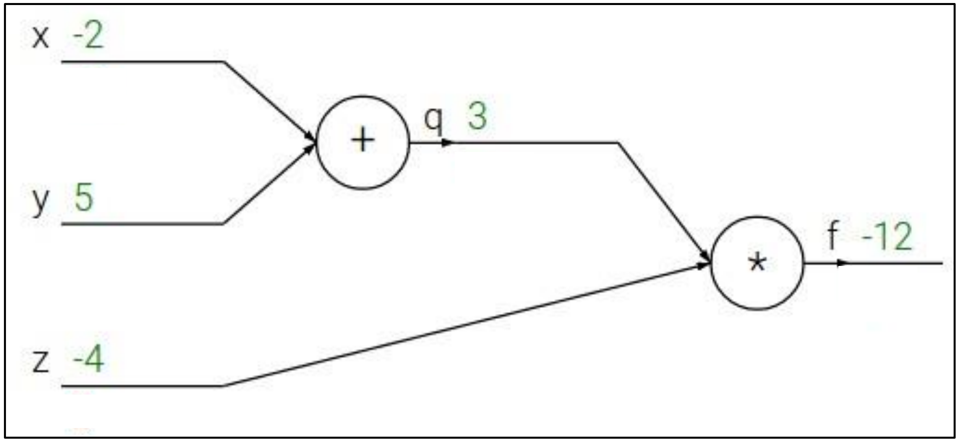
$$f(x, y, z) = (x + y)z$$

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



KYONGGI UNIVERSITY

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (3)

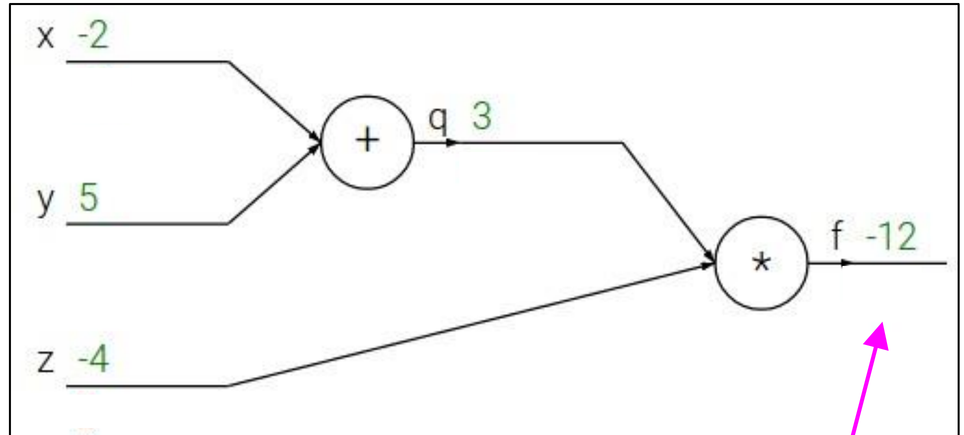
$$f(x, y, z) = (x + y)z$$

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

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

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

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



$$\boxed{\frac{\partial f}{\partial f}}$$

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (4)

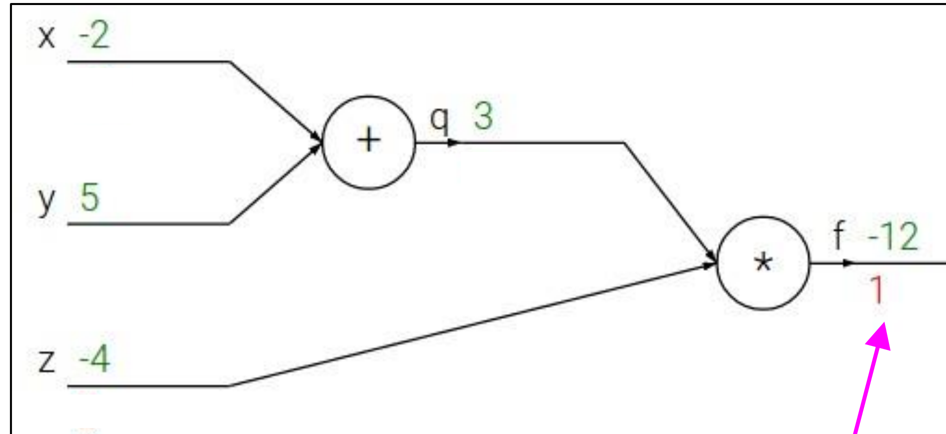
$$f(x,y,z) = (x+y)z$$

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



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

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (5)

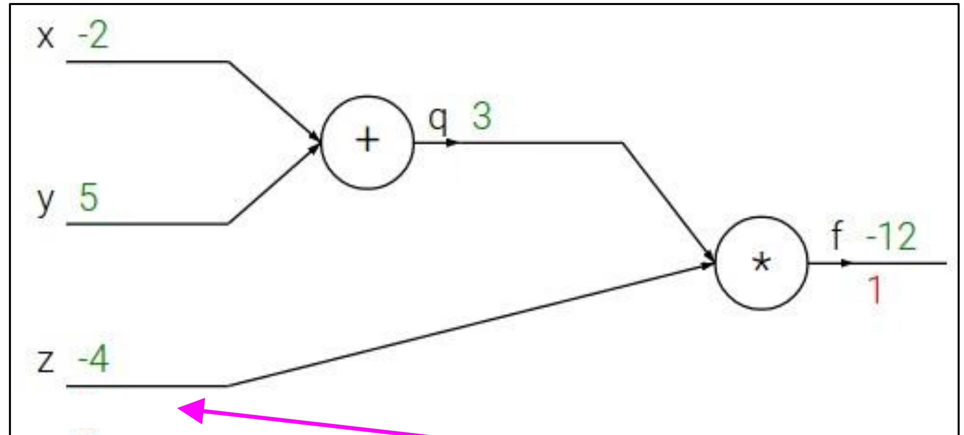
$$f(x, y, z) = (x + y)z$$

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



$$\frac{\partial f}{\partial z}$$

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (6)

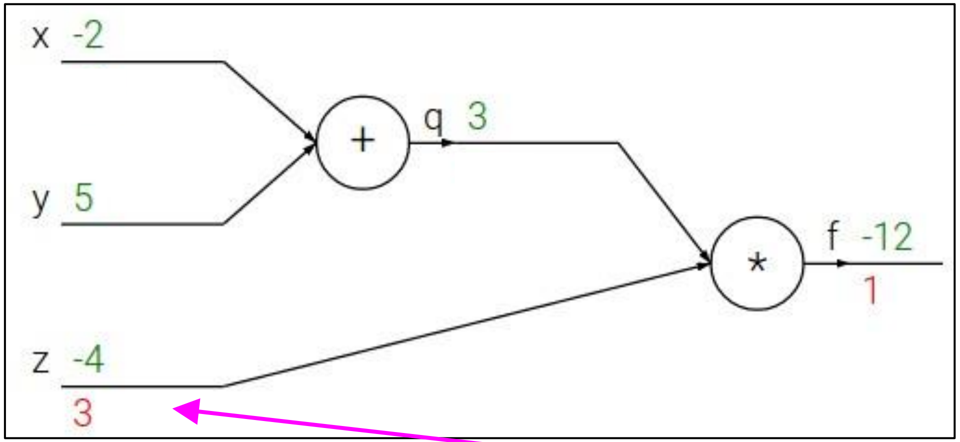
$$f(x, y, z) = (x + y)z$$

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



$$\frac{\partial f}{\partial z}$$

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (7)

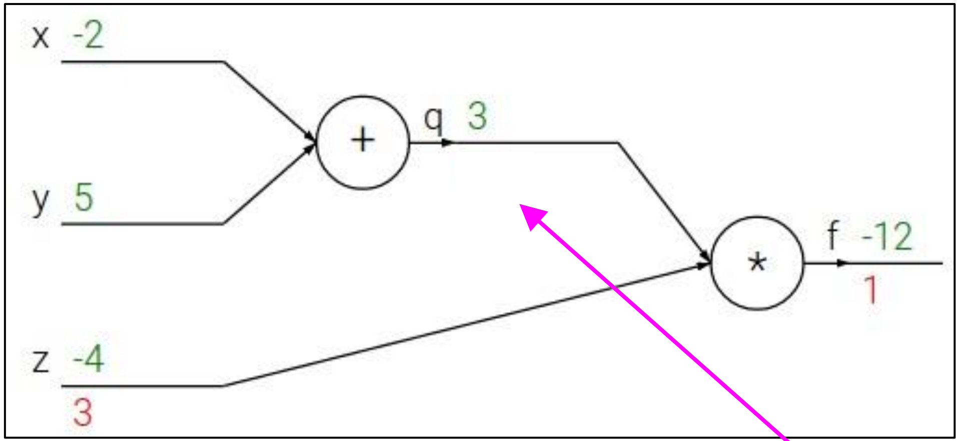
$$f(x, y, z) = (x + y)z$$

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



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

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (8)

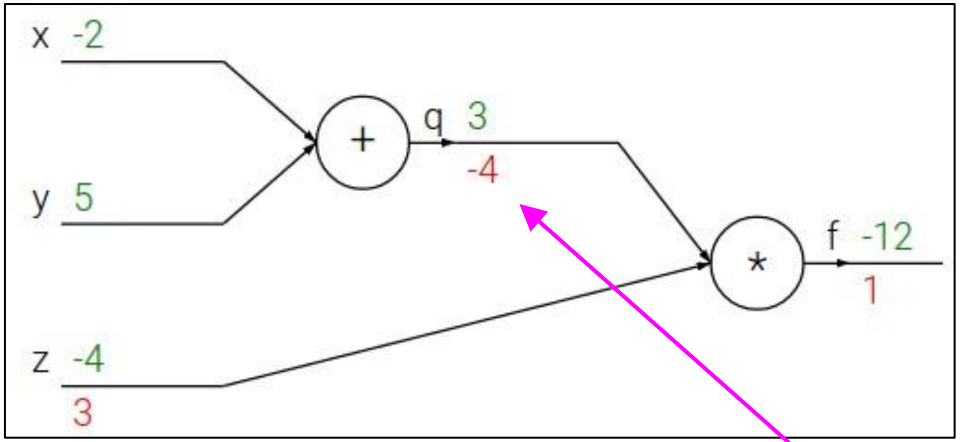
$$f(x, y, z) = (x + y)z$$

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



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

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (9)

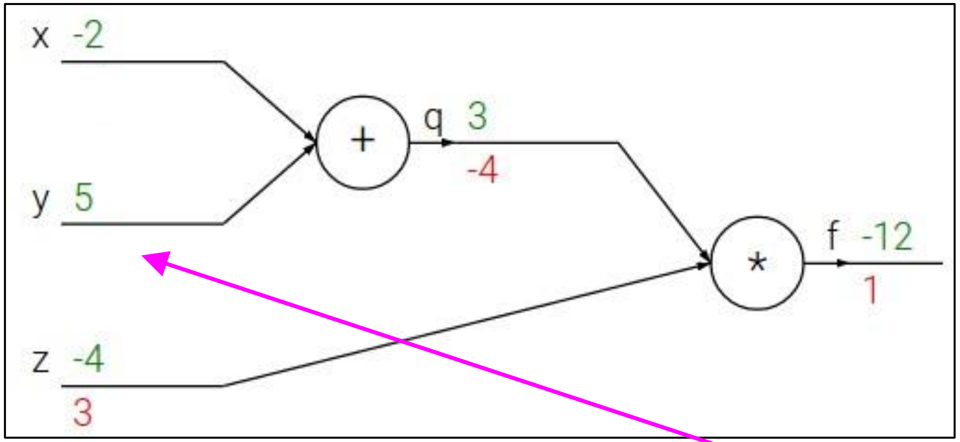
$$f(x, y, z) = (x + y)z$$

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



$$\boxed{\frac{\partial f}{\partial y}}$$

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (10)

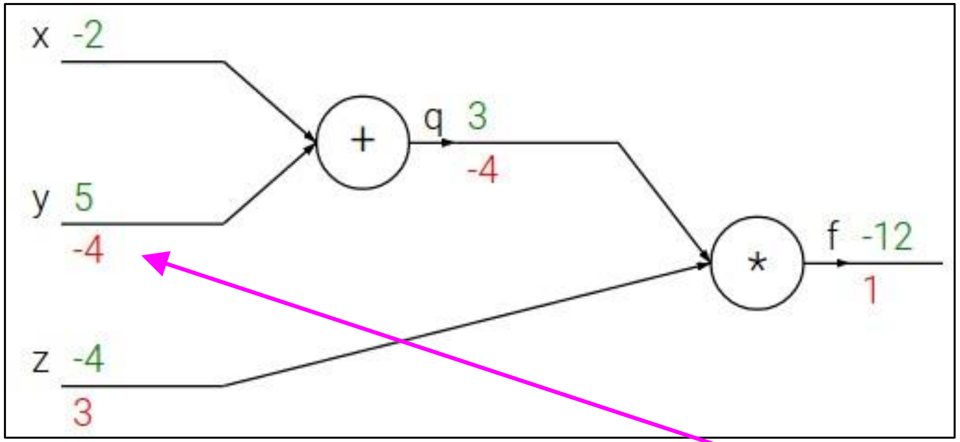
$$f(x, y, z) = (x + y)z$$

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



$$\frac{\partial f}{\partial y}$$

Chain rule:

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

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (11)

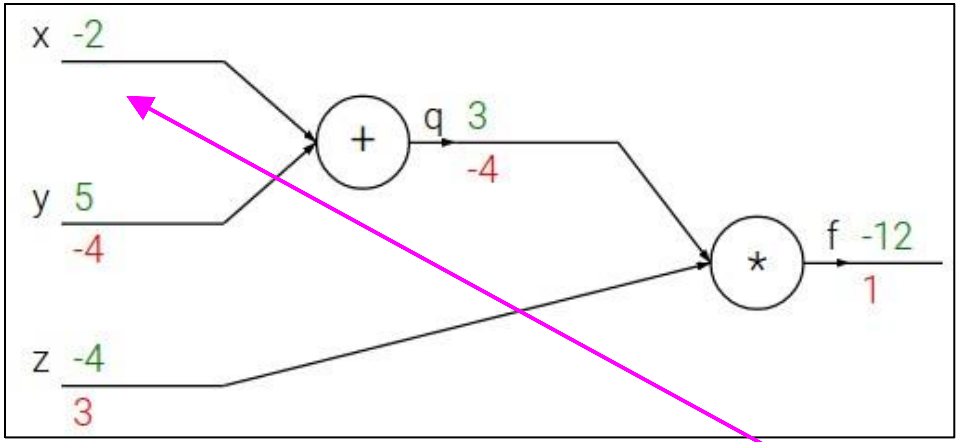
$$f(x, y, z) = (x + y)z$$

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



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

KYONGGI UNIVERSITY

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (12)

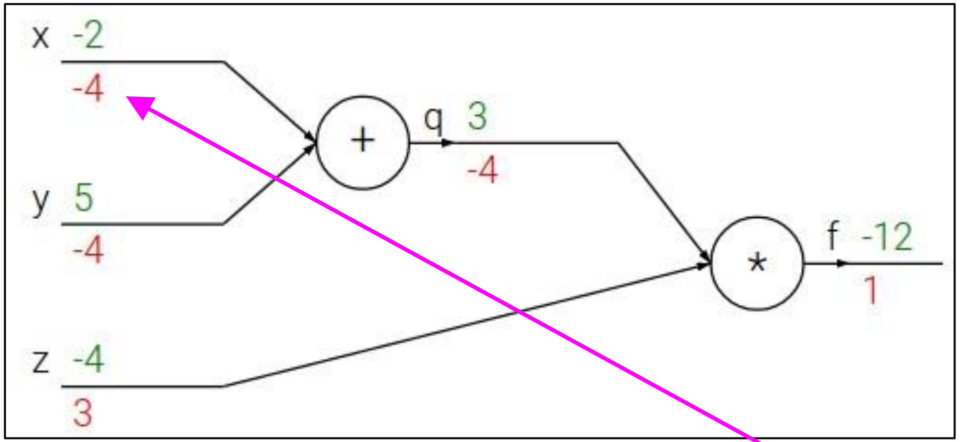
$$f(x, y, z) = (x + y)z$$

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

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

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

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



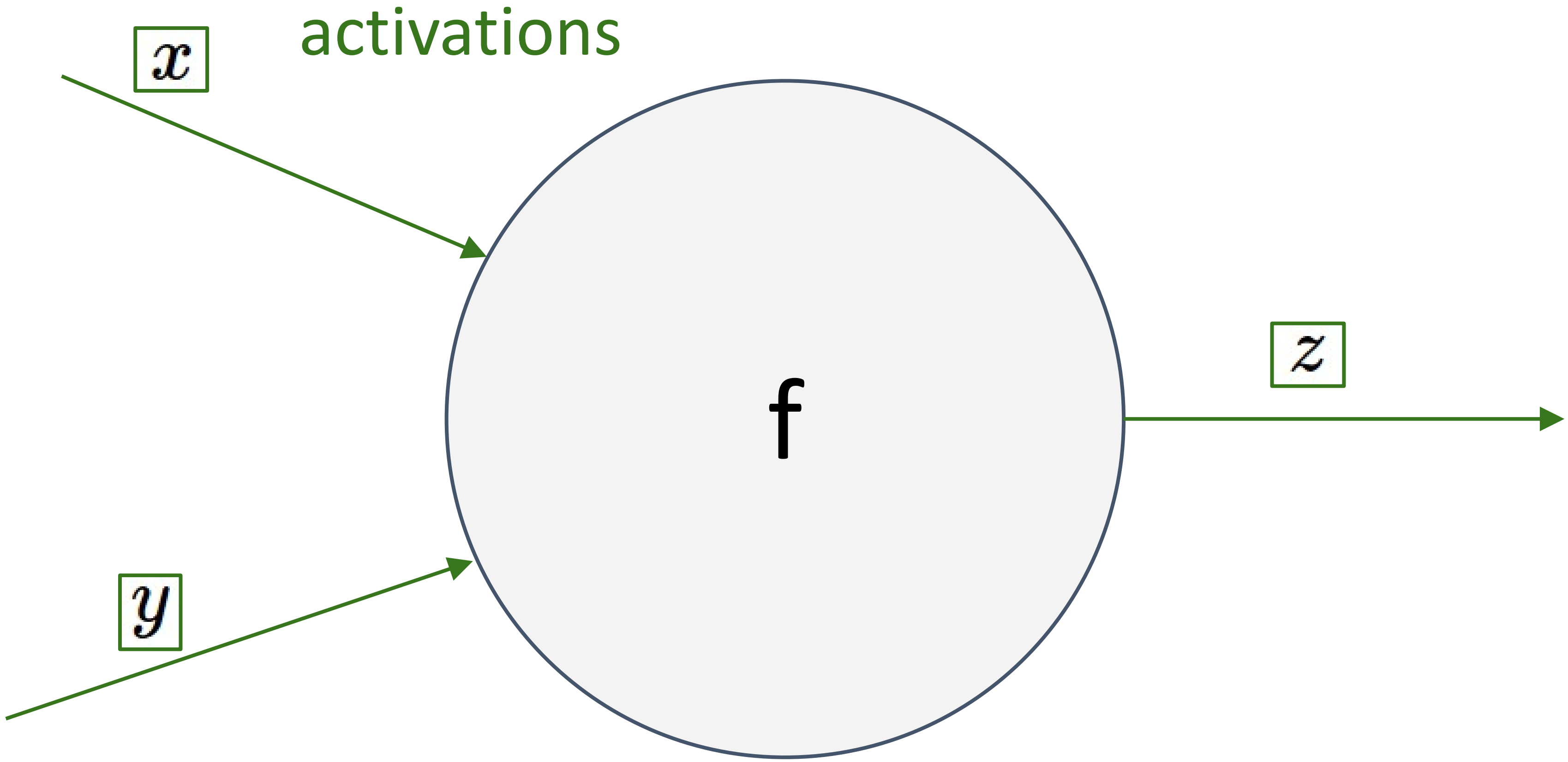
$\frac{\partial f}{\partial x}$

Chain rule:

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

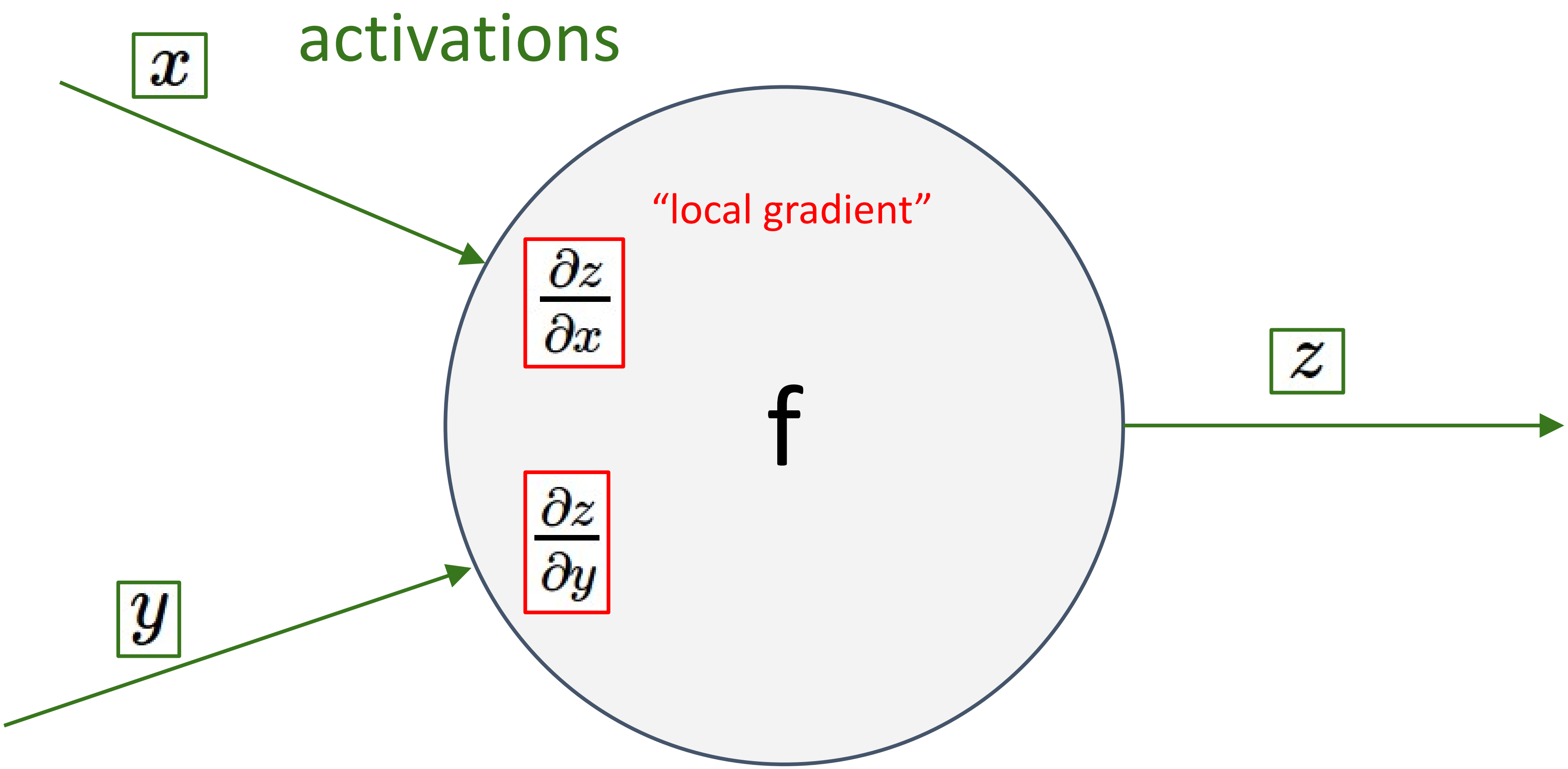
* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (13)



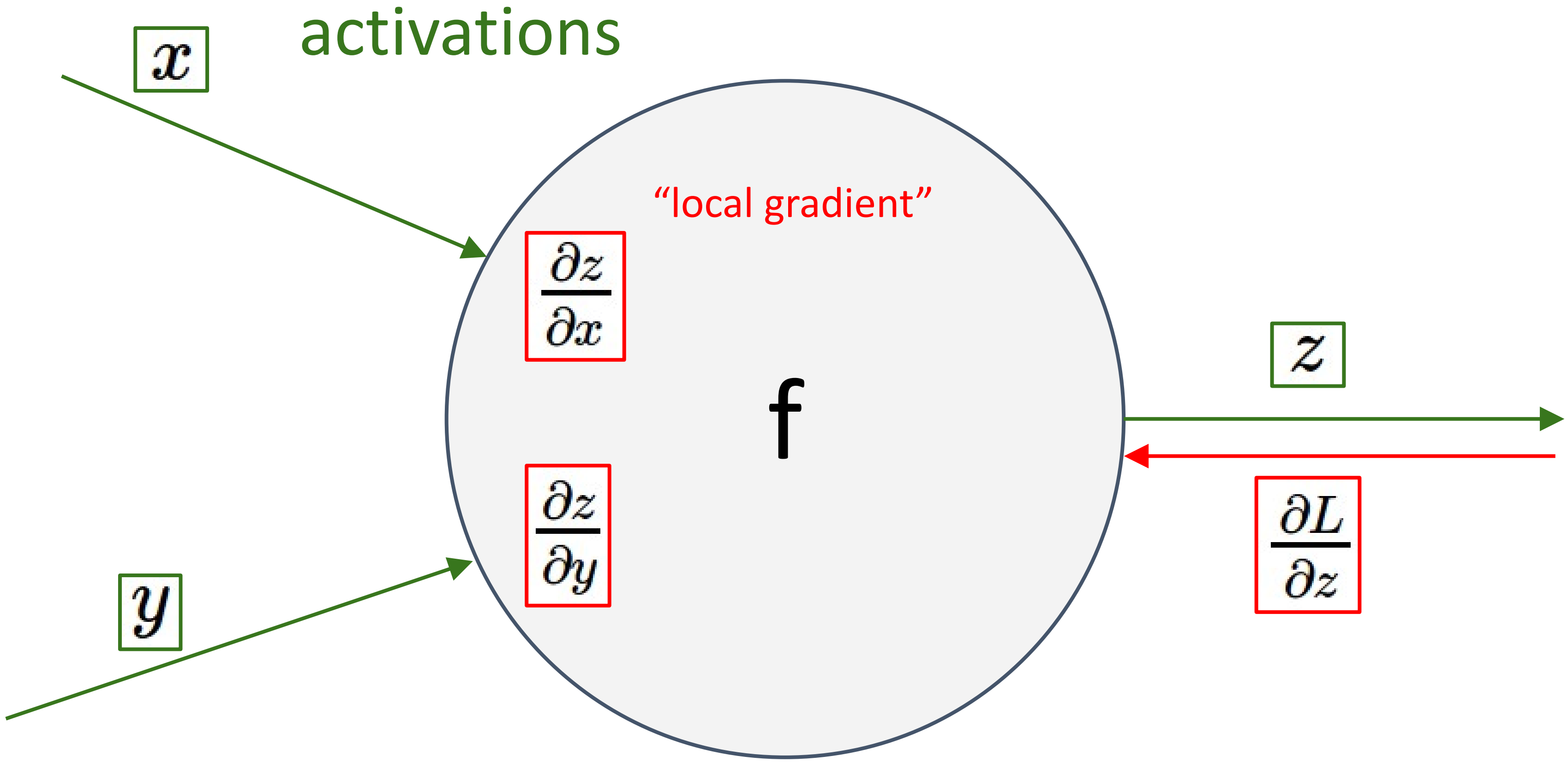
* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (14)



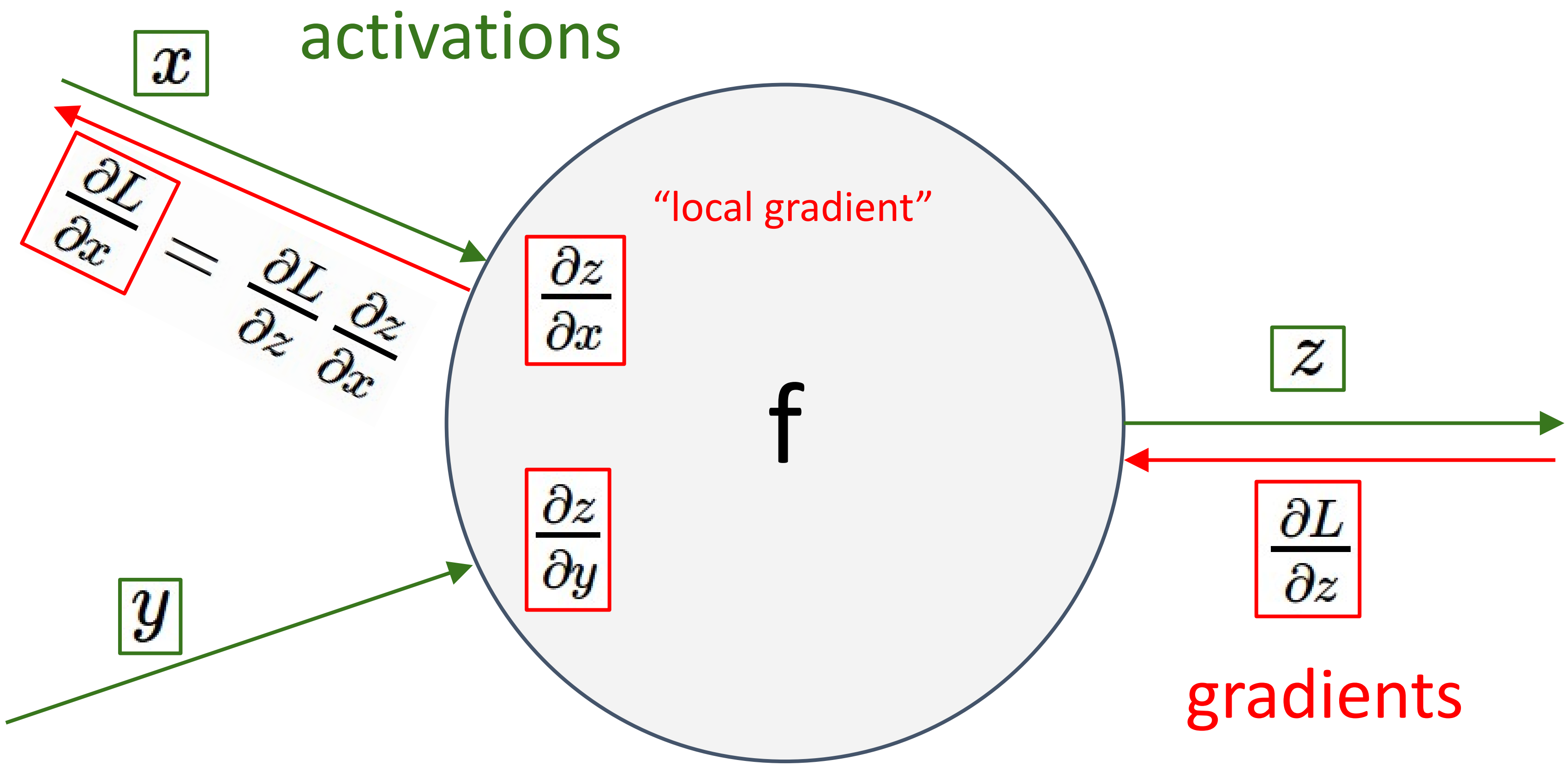
* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (15)



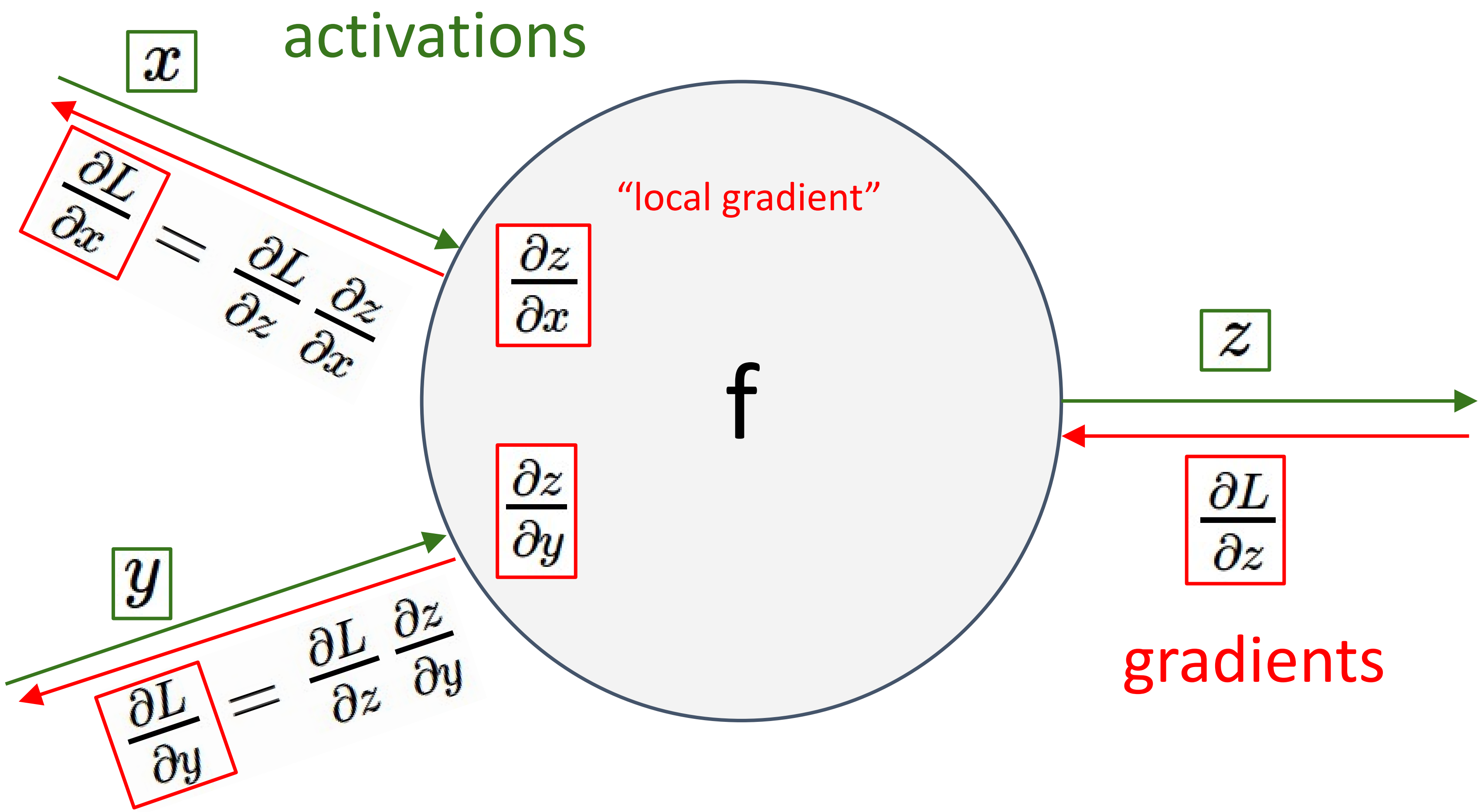
* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (16)



* Source: Stanford Univ. CS231 Lecture Notes

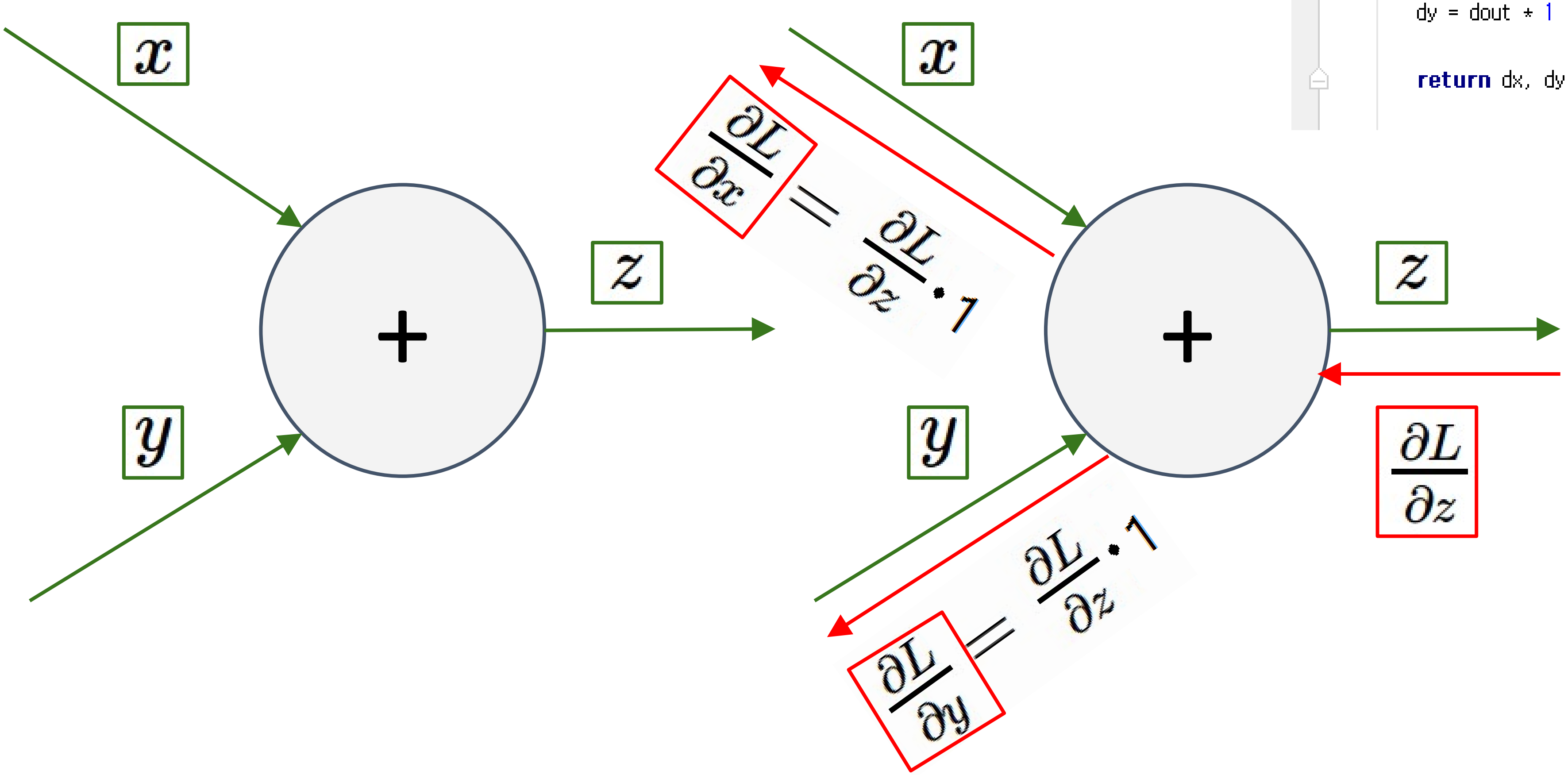
Backpropagation (17)



* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (18)

```python
class AddLayer:
    def __init__(self):
        pass

    def forward(self, x, y):
        out = x + y

        return out

    def backward(self, dout):
        dx = dout * 1
        dy = dout * 1

        return dx, dy
```

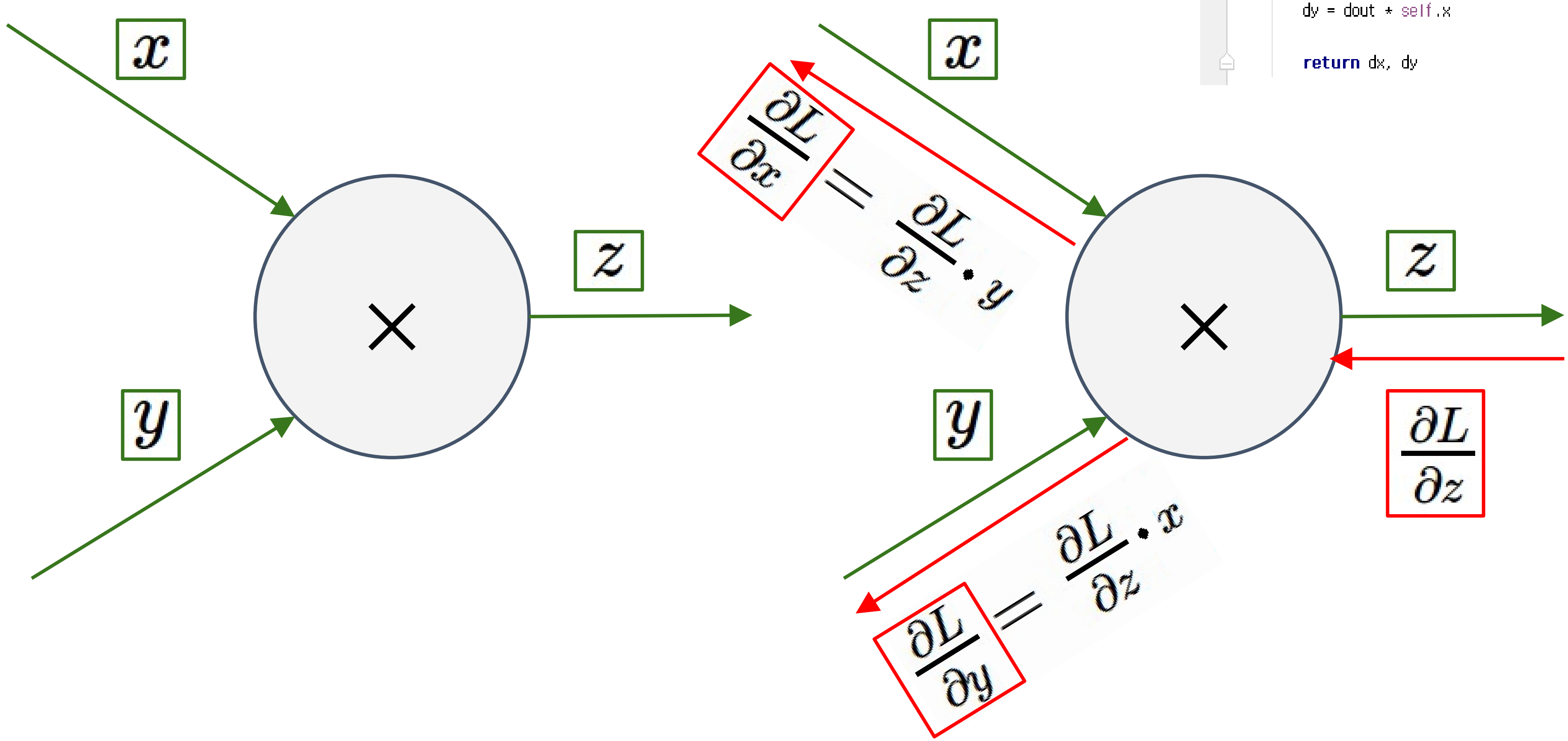


KYONGGI UNIVERSITY

* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (19)

```python
class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y):
        self.x = x
        self.y = y
        out = x * y

        return out

    def backward(self, dout):
        dx = dout * self.y  # x와 y를 바꾼다.
        dy = dout * self.x

        return dx, dy
```

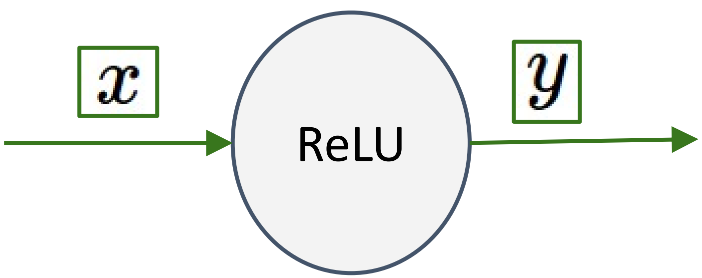


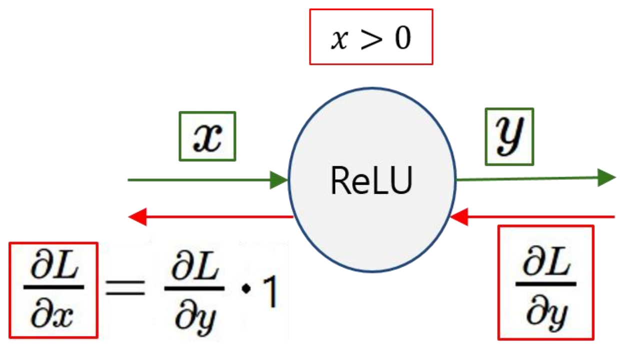
Backpropagation (20)

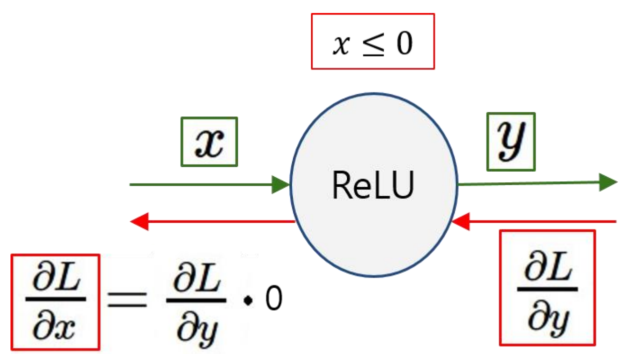
- ReLU

$$y = \begin{cases} x \ (x > 0) \\ 0 \ (x \le 0) \end{cases} \rightarrow \frac{\partial y}{\partial x} = \begin{cases} 1 \ (x > 0) \\ 0 \ (x \le 0) \end{cases}$$

```python
class ReLU:
    def __init__(self):
        self.mask = None

    def forward(self, x):
        self.mask = (x <= 0)
        out = x.copy()
        out[self.mask] = 0

        return out

    def backward(self, dout):
        dout[self.mask] = 0
        dx = dout

        return dx
```



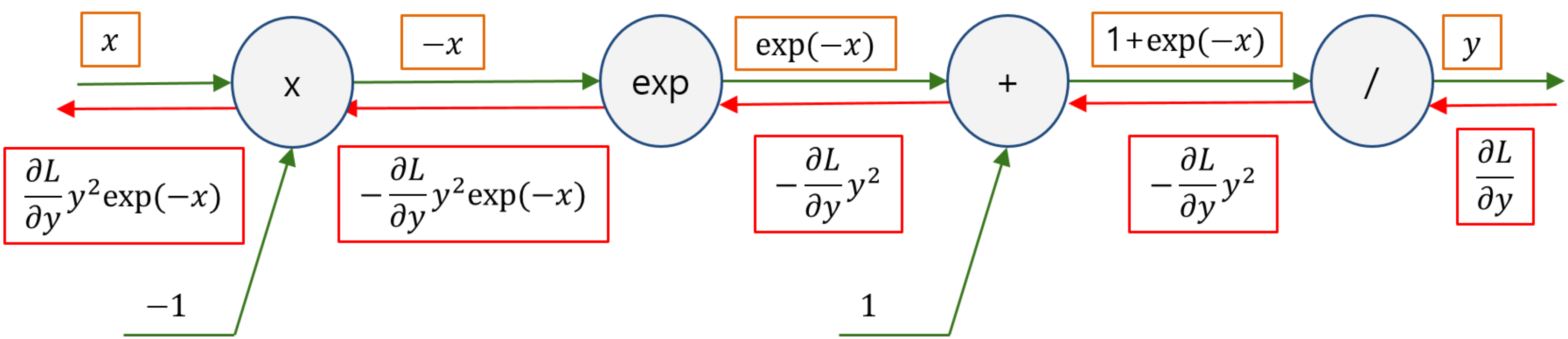




Backpropagation (21)

- Sigmoid

$$y = \frac{1}{1+\exp(-x)} \quad \rightarrow \quad \left(y = \frac{1}{x}, \frac{\partial y}{\partial x} = -\frac{1}{x^2} = -y^2\right), \left(y = \exp(x), \frac{\partial y}{\partial x} = \exp(x)\right)$$



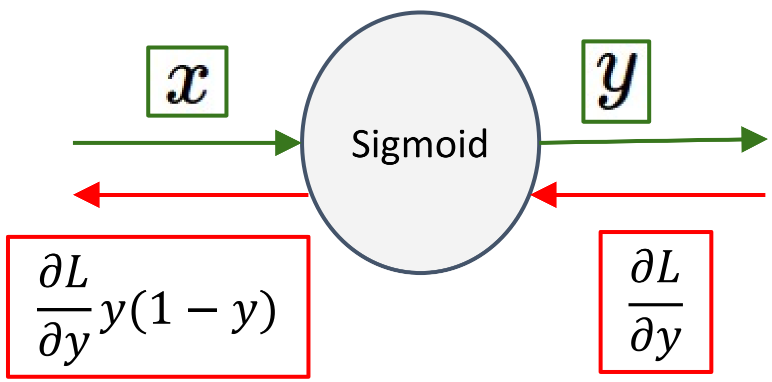
* Source: Stanford Univ. CS231 Lecture Notes

Backpropagation (22)

- Sigmoid (cont.)

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

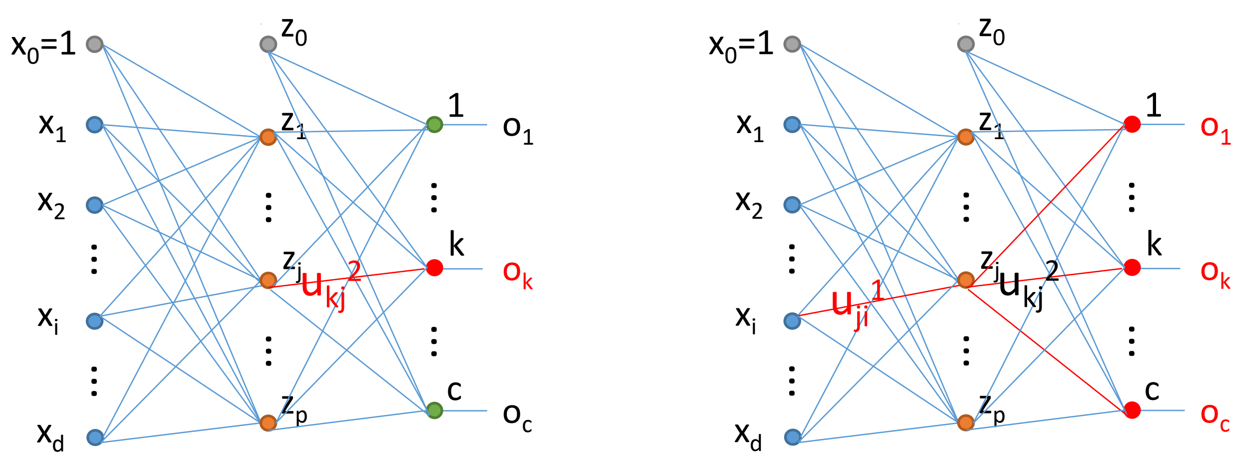
$$\frac{\partial L}{\partial y} y^2 \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)} = \frac{\partial L}{\partial y} y(1-y)$$




```python
class Sigmoid:
    def __init__(self):
        self.out = None

    def forward(self, x):
        out = 1 / (1+np.exp(-x))
        self.out = out

        return out

    def backward(self, dout):
        dx = dout * (1.0 - self.out)*self.out

        return dx
```

Backpropagation Example (1)

- 2-layer ANN
  - MSE Loss function: $E(\theta) = \frac{1}{2}\sum_k (y_k - t_k)^2$
  - 입력 차원 (d), 부류 개수 (c), 은닉층 노드 수 (p), one-hot encoding
  - Layer 1 가중치 행렬: U^1 ($p \times d$), Layer 2 가중치 행렬: U^2 ($c \times p$)

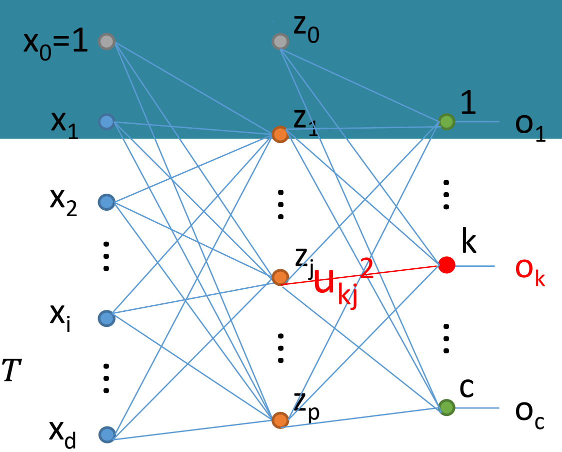


(a) u_{kj}^2가 미치는 영향 (b) u_{ji}^1이 미치는 영향

KYONGGI UNIVERSITY

* Source: 기계학습 (오일석), 한빛아카데미

Backpropagation Example (2)



$$o_k = \tau(osum_k), k = 1, 2, \dots, c$$

$$osum_k = u_k^2 \times z,\ u_k^2 = \left(u_{k0}^2, u_{k1}^2, \dots, u_{kp}^2\right), z = (1, z_1, z_2, \dots, z_p)^T$$

$$\frac{\partial E}{\partial u_{kj}^2} = \frac{\partial(0.5\|y - f(U^1, U^2)\|_2^2)}{\partial u_{kj}^2}$$

$$= \frac{\partial(0.5\sum_{q=1}^{c}(y_q - o_q)^2)}{\partial u_{kj}^2} = \frac{\partial(0.5(y_k - o_k)^2)}{\partial u_{kj}^2}$$

$$= -(y_k - o_k)\frac{\partial(o_k)}{\partial u_{kj}^2} = -(y_k - o_k)\frac{\partial\tau(osum_k)}{\partial u_{kj}^2}$$

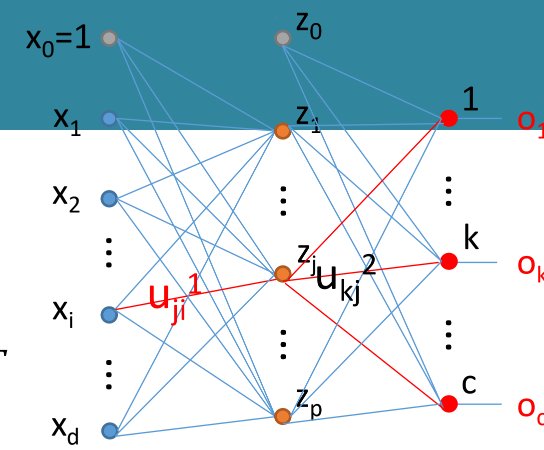
$$= -(y_k - o_k)\tau'(osum_k)\frac{\partial(osum_k)}{\partial u_{kj}^2}$$

$$= -(y_k - o_k)\tau'(osum_k)z_j$$

$$\boxed{\begin{aligned} &\delta_k = (y_k - o_k)\tau'(osum_k), 1 \le k \le c \\ &\frac{\partial E}{\partial u_{kj}^2} = -\delta_k z_j, 0 \le j \le p, 1 \le k \le c \end{aligned}}$$

* Source: 기계학습 (오일석), 한빛아카데미

Backpropagation Example (3)



$$z_j = \tau(zsum_j), j = 1, 2, \dots, p$$

$$zsum_j = u_j^1 \times x,\ u_j^1 = (u_{j0}^1, u_{j1}^1, \dots, u_{jd}^1),\ x = (1, x_1, x_2, \dots, x_d)^T$$

$$\frac{\partial E}{\partial u_{ji}^1} = \frac{\partial(0.5\|y - f(U^1, U^2)\|_2^2)}{\partial u_{ji}^1}$$

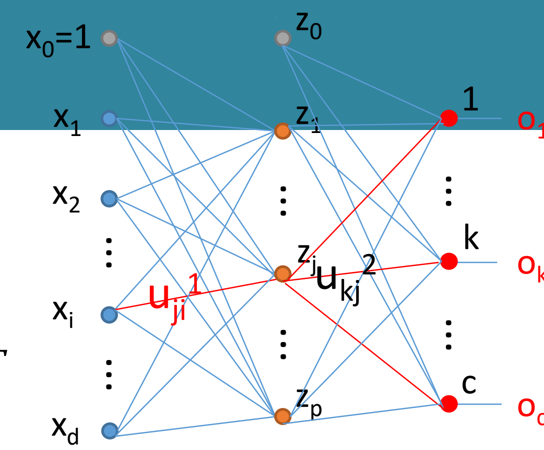
$$= \frac{\partial(0.5\sum_{q=1}^{c}(y_q - o_q)^2)}{\partial u_{ji}^1} = -\sum_{q=1}^{c}(y_q - o_q)\frac{\partial(o_q)}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q - o_q)\,\tau'(osum_q)\frac{\partial(osum_q)}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q - o_q)\,\tau'(osum_q)\frac{\partial(osum_q)}{\partial z_j}\frac{\partial(z_j)}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q - o_q)\,\tau'(osum_q)u_{qj}^2\frac{\partial(z_j)}{\partial u_{ji}^1}$$

Backpropagation Example (3)



$$z_j = \tau(zsum_j), j = 1, 2, \dots, p$$

$$zsum_j = u_j^1 \times x,\ u_j^1 = (u_{j0}^1, u_{j1}^1, \dots, u_{jd}^1),\ x = (1, x_1, x_2, \dots, x_d)^T$$

$$= -\sum_{q=1}^{c} (y_q - o_q)\,\tau'(osum_q) u_{qj}^2 \frac{\partial (z_j)}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c} (y_q - o_q)\,\tau'(osum_q) u_{qj}^2 \tau'(zsum_j) x_i$$

$$= -\tau'(zsum_j) x_i \underbrace{\sum_{q=1}^{c} (y_q - o_q)\,\tau'(osum_q) u_{qj}^2}_{\delta_q}$$

$$\boxed{\begin{aligned} \delta_k &= (y_k - o_k)\tau'(osum_k), 1 \le k \le c \\ \frac{\partial E}{\partial u_{kj}^2} &= -\delta_k z_j, 0 \le j \le p, 1 \le k \le c \end{aligned}}$$

$$\boxed{\begin{aligned} \eta_j &= \tau'(zsum_j) \sum_{q=1}^{c} \delta_q u_{qj}^2, 1 \le j \le p \\ \frac{\partial E}{\partial u_{ji}^1} &= -\eta_j x_i, 0 \le i \le d, 1 \le j \le p \end{aligned}}$$

Training Process (1)

- 배치 학습 (batch learning)
  - 모든 훈련 샘플에 대해 계산되는 $E_n(w)$의 합을 오차함수로 사용

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

- 확률적 경사 하강법 (SGD: Stochastic Gradient Descent)
  - 샘플의 일부, 극단적으로는 샘플 하나만을 사용하여 가중치 업데이트 수행
  - DNN에서는 배치 학습보다는 SGD를 더 많이 사용

Training Process (2)

▪ Minibatch

- 샘플 한 개 단위가 아니라 몇 개의 샘플을 하나의 작은 집합으로 묶은 단위 (e.g., minibatch)로 가중치 업데이트 수행

$$E(w) = \frac{1}{N_t}\sum_{n \in D_t}^{N} E_n(w)\,, N_t = |D_t|$$

- 일반적으로 $D_t(t = 1, 2, \ldots)$는 학습 시작 전에 결정하여 고정
  - 미니배치의 크기는 대체로 10~100개 샘플 전후로 결정하는 경우가 많음
  - 클래스의 수가 10~100개 정도이면서 클래스의 출현 빈도가 서로 같을 경우 분류하려는 클래스 수와 같은 크기의 미니배치를 생성하는 것이 좋음
  - 클래스 수가 그 이상인 경우 샘플을 랜덤으로 섞은 후에 미니배치를 분할

KYONGGI UNIVERSITY

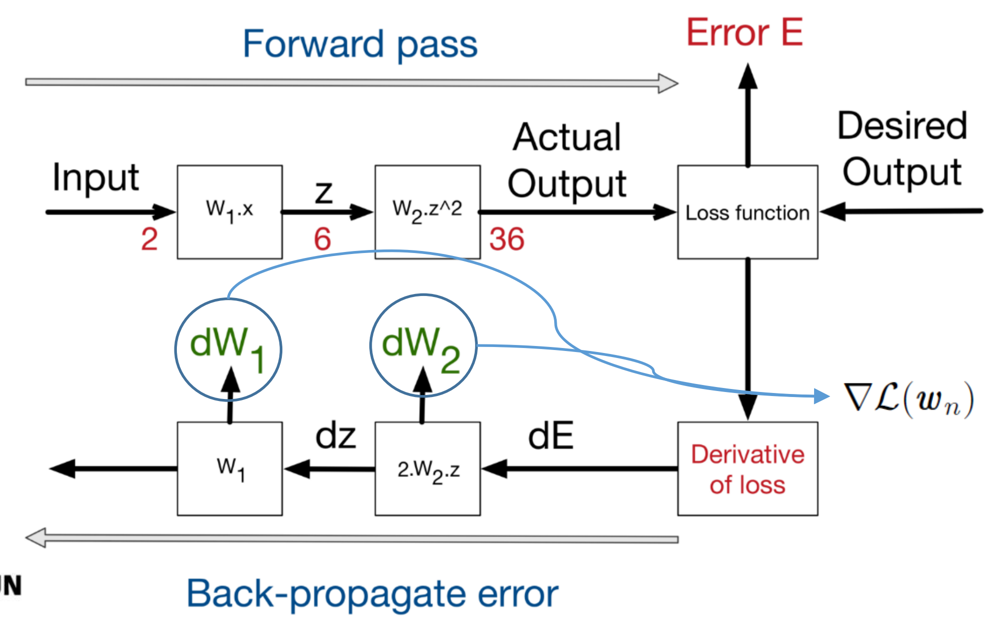
Training Process (3)

- The objective of training process

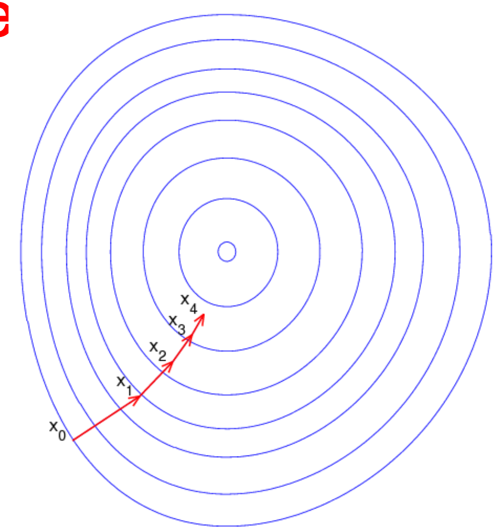
$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

- At each iteration, we need to compute

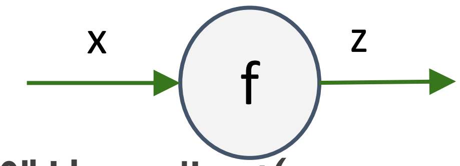
$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \gamma_n \nabla \mathcal{L}(\boldsymbol{w}_n)$$



$$\nabla \mathcal{L}(\boldsymbol{w}_n) = \frac{1}{N}\sum_{i=1}^{N} \nabla l(\boldsymbol{w}_n, \boldsymbol{x}_i, t_i) + \nabla \mathcal{R}(\boldsymbol{w}_n)$$

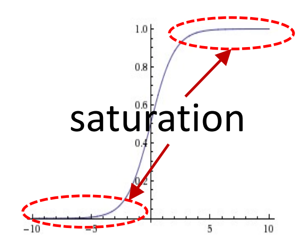


Vanishing Gradient Program (1)



- Vanishing Gradient란?
  - NN의 depth가 깊어지게 되면 $w_{n+1} = w_n - \alpha \times \nabla f(= \frac{\partial f}{\partial x})$ 에서 gradient(e.g., $\frac{\partial f}{\partial x}$)가 아주 작은 값을 가지게 되며, 따라서 Input Layer에 가까운 Layer일 수록 w에 변화가 거의 없어지게 됨
- Why does it happen?
  - $\frac{\partial L}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial L}{\partial z}$가 되며 이 중 하나의 항이라도 0에 가까울수록 $\frac{\partial L}{\partial x}$는 0에 가까워지며 결과적으로 $\alpha \times \nabla f$ 이 0에 가까워짐에 따라 w_{n+1}과 w_n이 변화가 없게 됨
  - Sigmoid 함수를 사용하게 될 경우 함수가 saturation이 되는 부분에서는 기울기의 변화가 거의 없으며 따라서 $\frac{\partial z}{\partial x}$(e.g., local gradient)가 0에 가깝게 됨
  - Output Layer에서 가까운 neuron이 항상 1에 가까운 값을 낼 경우, 즉 Saturation 될 경우 그 neuron과 연결된 앞쪽 Layer에서 사용되는 w는 거의 변화가 없어지게 됨

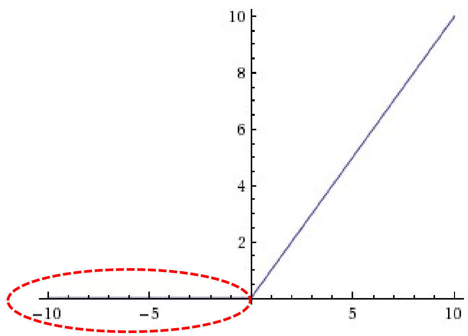
  ➔ Error의 Backpropagation이 제대로 이루어지지 않음



Vanishing Gradient Problem (2)

ReLU 함수를 사용할 경우

- ReLU는 x가 0보다 큰 경우 선형적으로 증가하기 때문에 $\frac{\partial z}{\partial x}$(e.g., local gradient)가 0이 되는 경우는 발생하지 않음. 따라서 gradient가 0에 근접하는 경우는 발생하지 않음
- 그러나 특정 neuron이 선택되지 않는 경우 (즉, ReLU의 output이 0이 발생하는 구간)에는 local gradient, 즉 $\frac{\partial z}{\partial x} = 0$이 됨. 따라서 해당 neuron과 연결된 앞쪽 neuron의 경우 gradient가 항상 0이 되기 때문에 w에 변화가 발생하지 않음



Parameter Updates (1)

- SGD

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

- Momentum

$$v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W}$$

$$W \leftarrow W + v$$

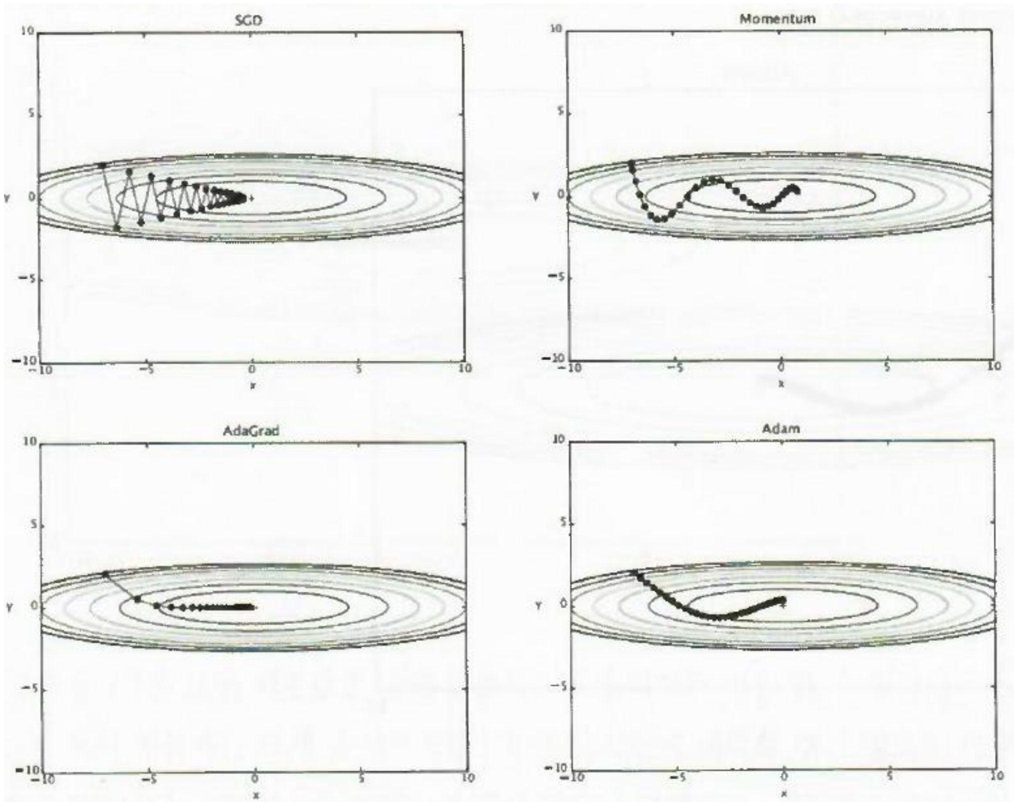
- AdaGrad
  - 각각의 매개변수를 대상으로 학습률 감소(learning rate decay) 적용

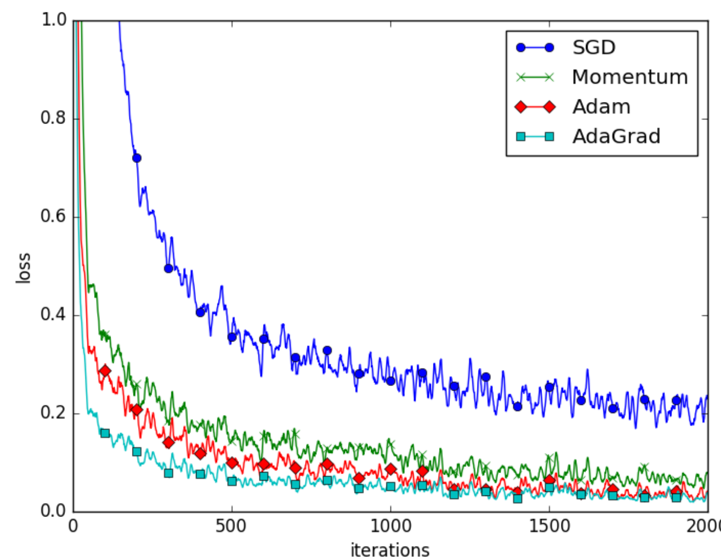
$$h \leftarrow h + \frac{\partial L}{\partial W} \cdot \frac{\partial L}{\partial W}$$

$$W \leftarrow W + \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

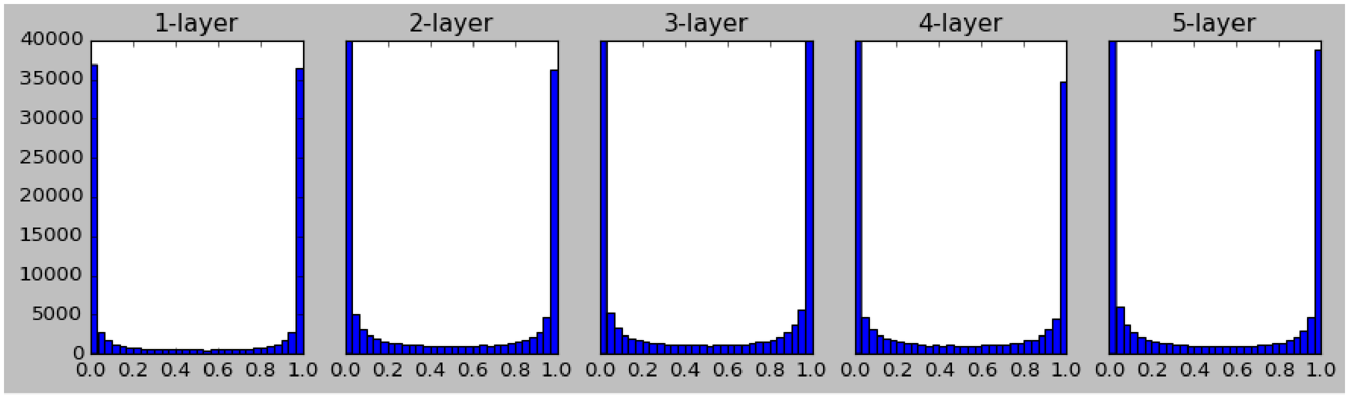
- Adam
  - Momentum + AdaGrad

Parameter Updates (2)

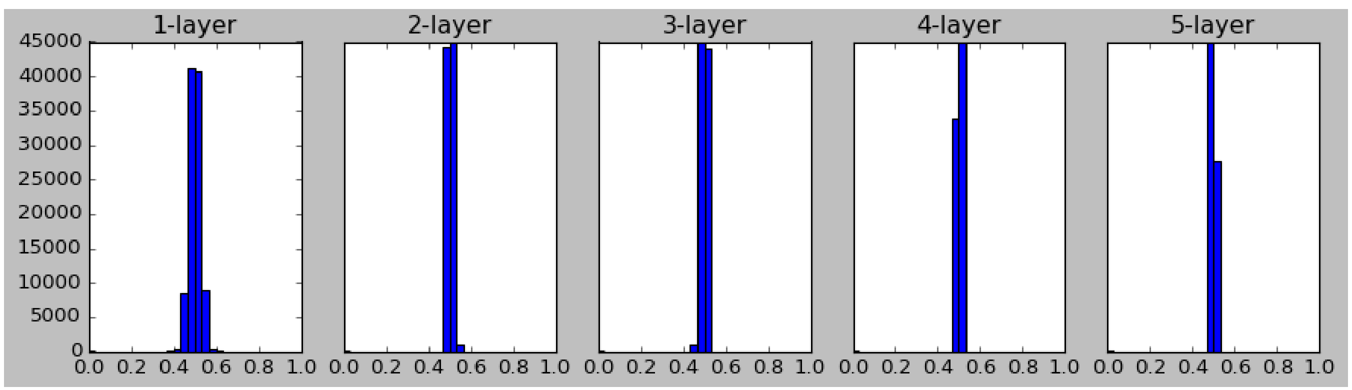




Initialization (1)



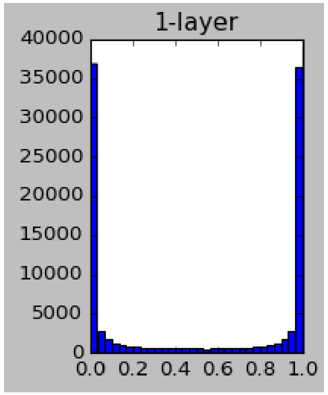
가중치를
표준편차가 1인
정규분포로
초기화할 때의 각
층의 활성화값
분포



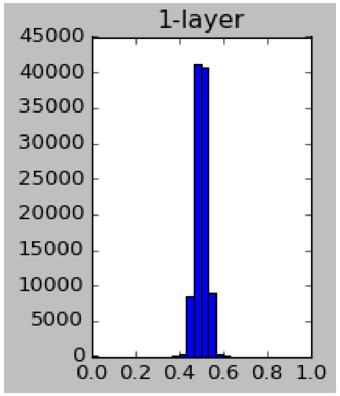
가중치를
표준편차가 0.01인
정규분포로
초기화할 때의 각
층의 활성화값
분포

Initialization (2)

- **각 층의 활성화값 (활성화 함수의 출력 데이터)은 적당히 골고루 분포되어야 함**
  - 층과 층 사이에 적당하게 다양한 데이터가 흐르게 해야 신경망 학습이 효율적으로 이루어짐
  - 반대로 치우친 데이터가 흐르면 기울기 소실이나 표현력 제한 문제에 빠져 학습이 잘 이루어지지 않는 경우 발생



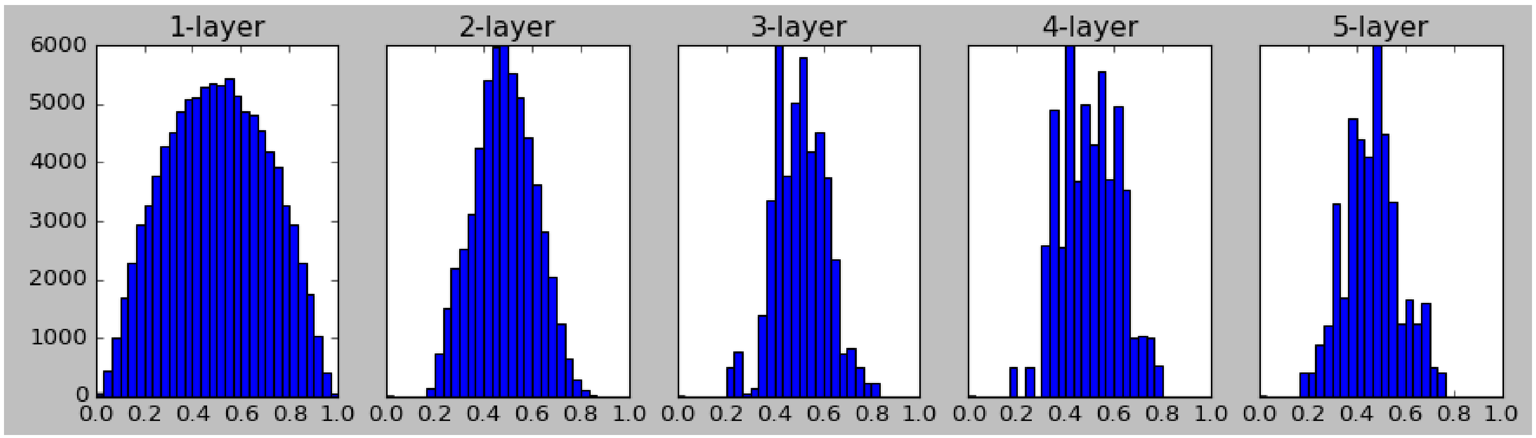
데이터가 0과 1에 치우쳐지게 분포
→ Sigmoid 함수의 Saturation Point
→ Vanishing Gradient 문제 발생



다수의 뉴런의 유사한 값 출력
→ 뉴런을 여러 개 둔 의미가 없어짐
(100개의 뉴런이 같은 값을 가질 경우
1개의 뉴런으로 구성하는 것과 동일

Initialization (3)

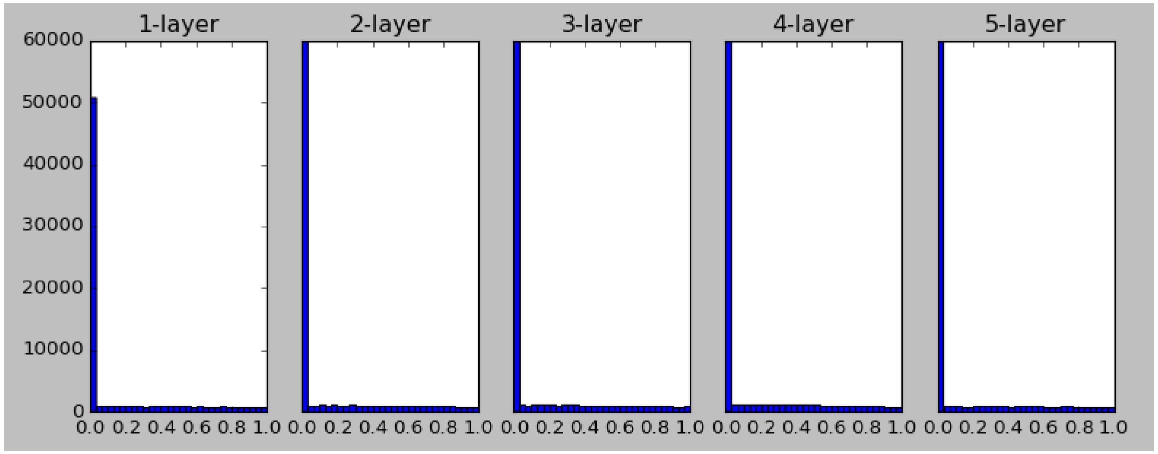
- Xavier Initialization
  - 표준편차가 ($^1/_{\sqrt{n}}$)인 정규 분포로 가중치 초기값 설정 (n: 앞 층의 노드 수)



  - Sigmoid, tanh 등 활성화 함수가 선형일 때 사용 (좌우 대칭 형태이므로 중앙 부근이 선형으로 볼 수 있음

Initialization (4)

- He Initialization
  - 표준편차가 ($1/\sqrt{n/2}$)인 정규 분포로 가중치 초기값 설정 (n: 앞 층의 노드 수)



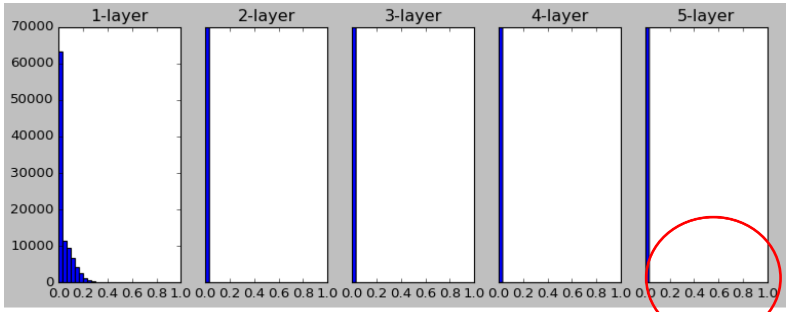
  - 활성화 함수로 ReLU 함수를 사용할 때 사용

Initialization (5)

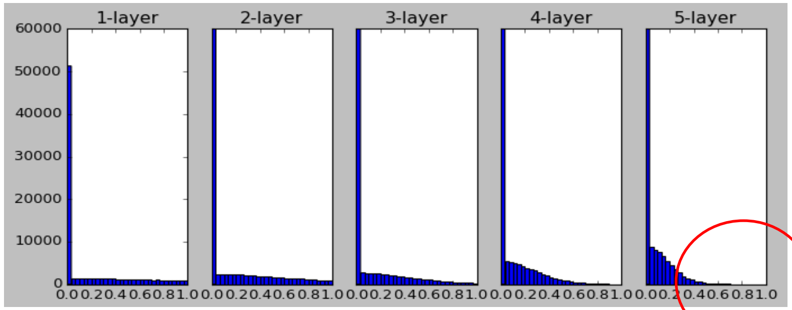
- Bias Initialization
  - Bias는 0으로 초기화하는 것이 보편적
  - ReLU를 사용할 경우 0.01과 같은 작은 값으로 초기화하면 성능 향상 효과가 있다는 보고도 있으나 모든 경우 그렇지는 않음

Initialization (6)

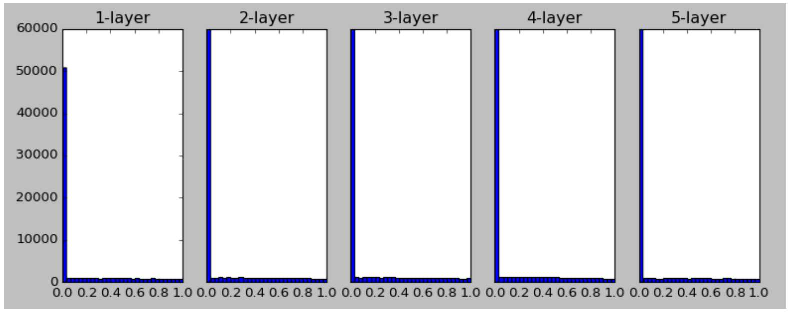
- 활성화 함수로 ReLU를 사용한 경우 가중치 초기값에 따른 활성화값 분포 변화



[표준편차가 0.01인 정규분포]()
→ 역전파 시 가중치의 기울기 역시 작아짐
→ 학습이 거의 이루어지지 않음



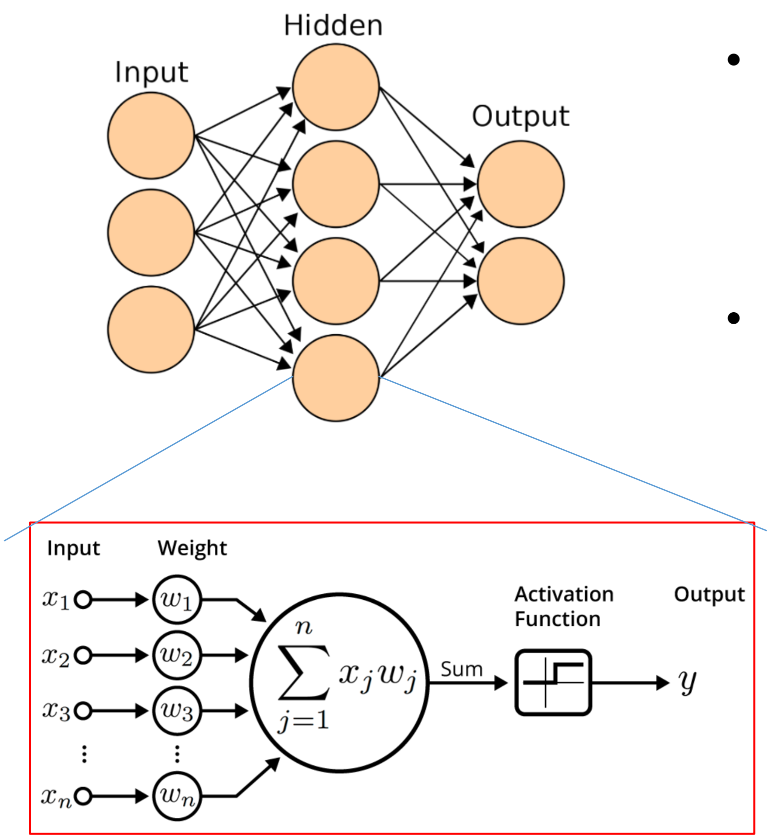
[Xavier 초기값]()
→ Vanishing Gradient 문제 발생 가능



[He 초기값]()
→ 층이 깊어져도 분포가 모든 층에서 균일하게 분포

Wrap-Up

■ Multi-Layer Perceptron



- 학습 목표 → Loss function

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

- 학습 방법 → Gradient Descent Method

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

References

- CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.stanford.edu/
- 라온피플 머신러닝 아카데미, https://laonple.blog.me/
- 밑바닥부터 시작하는 딥러닝, 한빛미디어
- 기초 수학으로 이해하는 머신러닝 알고리즘, 위키북스
- 딥러닝 제대로 시작하기, 제이펍
- 기계학습, 한빛아카데미


ANY
QUESTIONS?