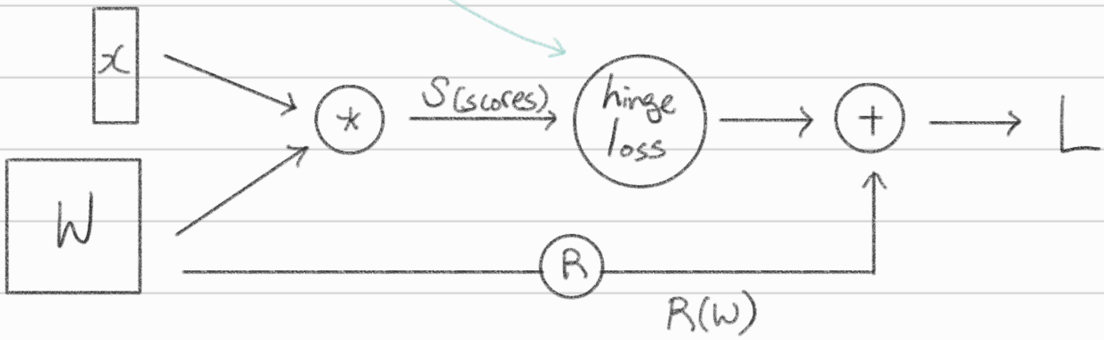
Backpropagation and Neural Networks *

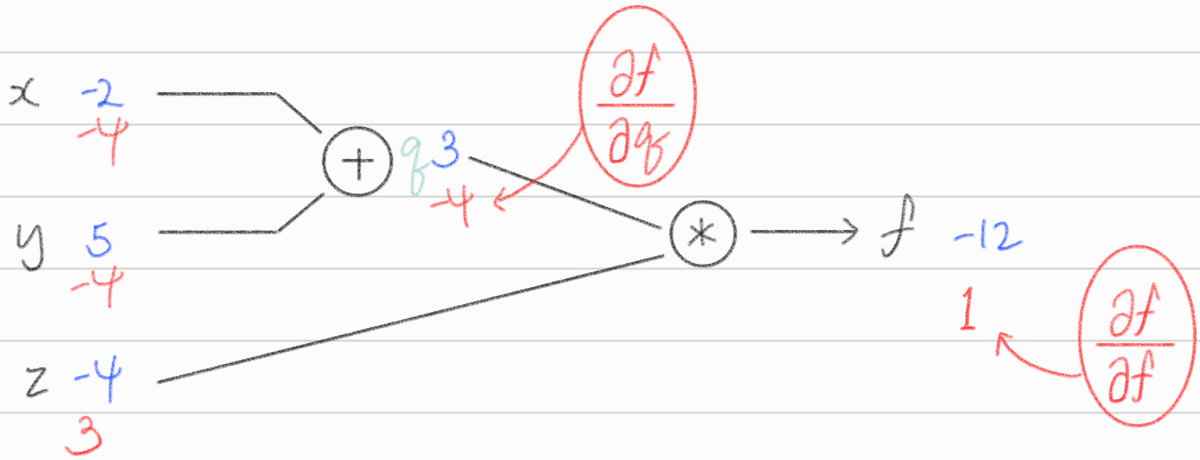
Computational graphs

$f = Wx$ $\quad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$



* Backpropagation 역전파

$f(x, y, z) = (x+y)z$ $\quad$ e.g. $x=-2, y=5, z=-4$



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

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

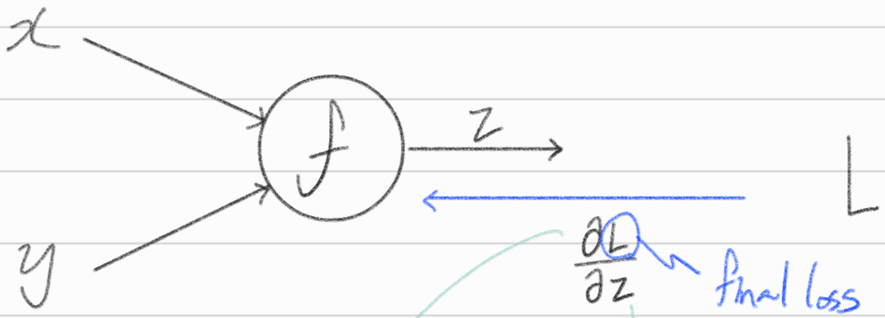
$f = qz \qquad \frac{\partial f}{\partial q} = z, \ \frac{\partial f}{\partial z} = q$

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

////// : gradient

$$\frac{\partial f}{\partial x} = \frac{\partial \cancel{q}}{\partial x} \cdot \frac{\partial f}{\partial \cancel{q}} = 1 \cdot z = z = -4$$

$$\frac{\partial f}{\partial y} = \frac{\partial \cancel{q}}{\partial y} \cdot \frac{\partial f}{\partial \cancel{q}} = 1 \cdot z = z = -4$$



local gradient $\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$ 계산 가능

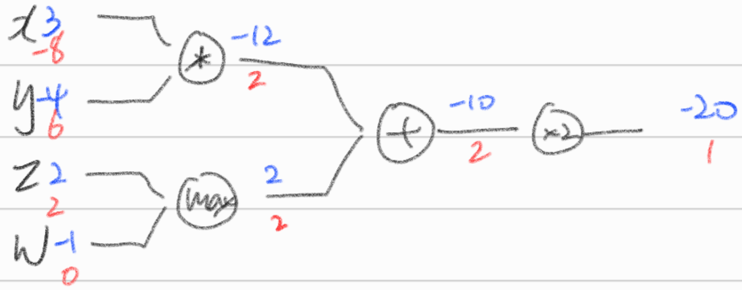
즉, Chain rule 통해 final loss 에 대한 x, y gradient 계산가능

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\cancel{\partial z}} \cdot \frac{\cancel{\partial z}}{\partial x} \qquad \frac{\partial L}{\partial y} = \frac{\partial L}{\cancel{\partial z}} \cdot \frac{\cancel{\partial z}}{\partial y}$$

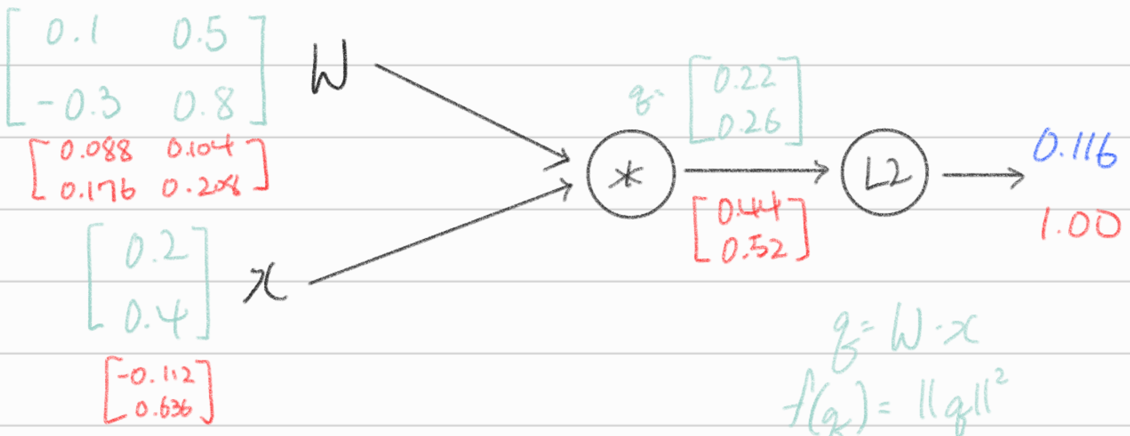
Patterns in backward flow

- add gate : gradient distributor
- max gate : gradient router
- mul gate : gradient switcher

$2(xy + \max(z, w))$



A vectorized ex. $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$ L2



$q = W \cdot x$

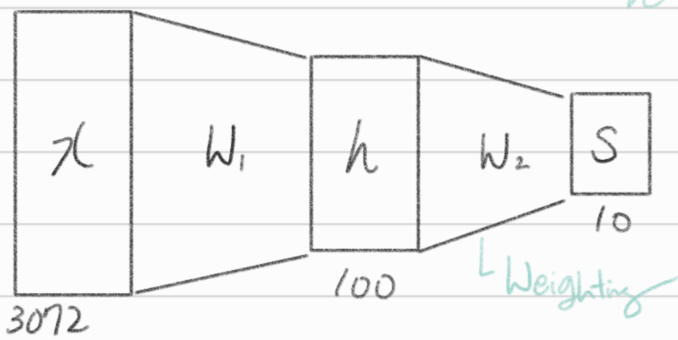
$f(q) = \|q\|^2$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

Neural networks

Linear score function : $f = Wx$

2-layer Neural Network : $f = W_2 \underbrace{\max(0, W_1 x)}_{h}$



- Arrange neurons into fully-connected layers
- Abstraction of a layer has the nice property that it allows us to use efficient vectorized code
- Neural networks are not really neural