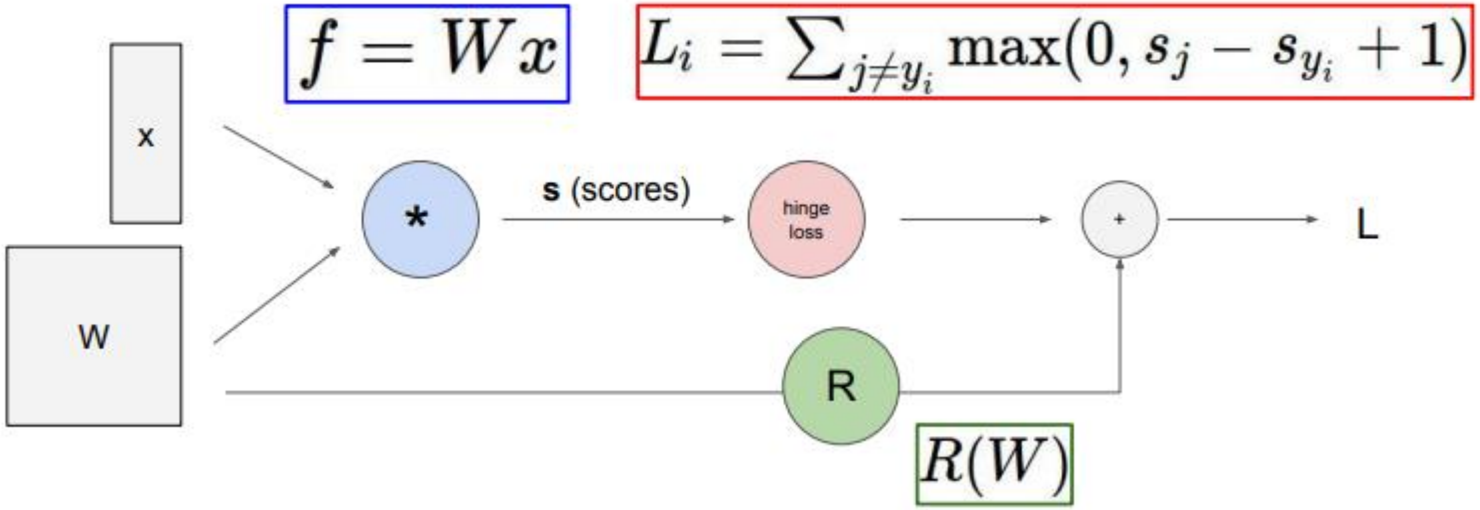
CS231n 4강
Backpropagation and Neural Networks

Computational graphs



계산하려는 과정(함수)을 일련의 간단한 그래프의 순서로 나타낸것이 **Computational graph**이다. 이를 보면서 계산하거나 로직을 이해하기 쉽다!

Backpropagation: a simple example

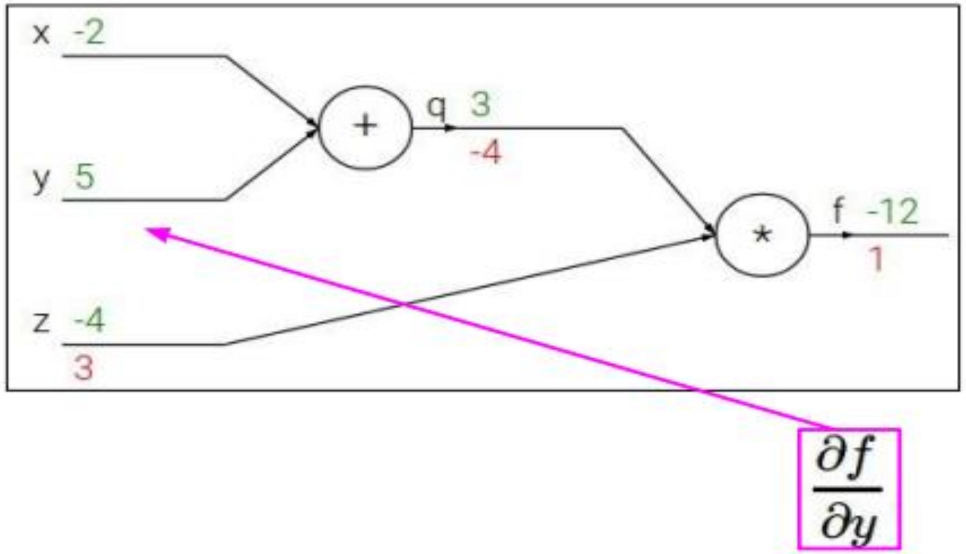
$$f(x, y, z) = (x + y)z$$

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

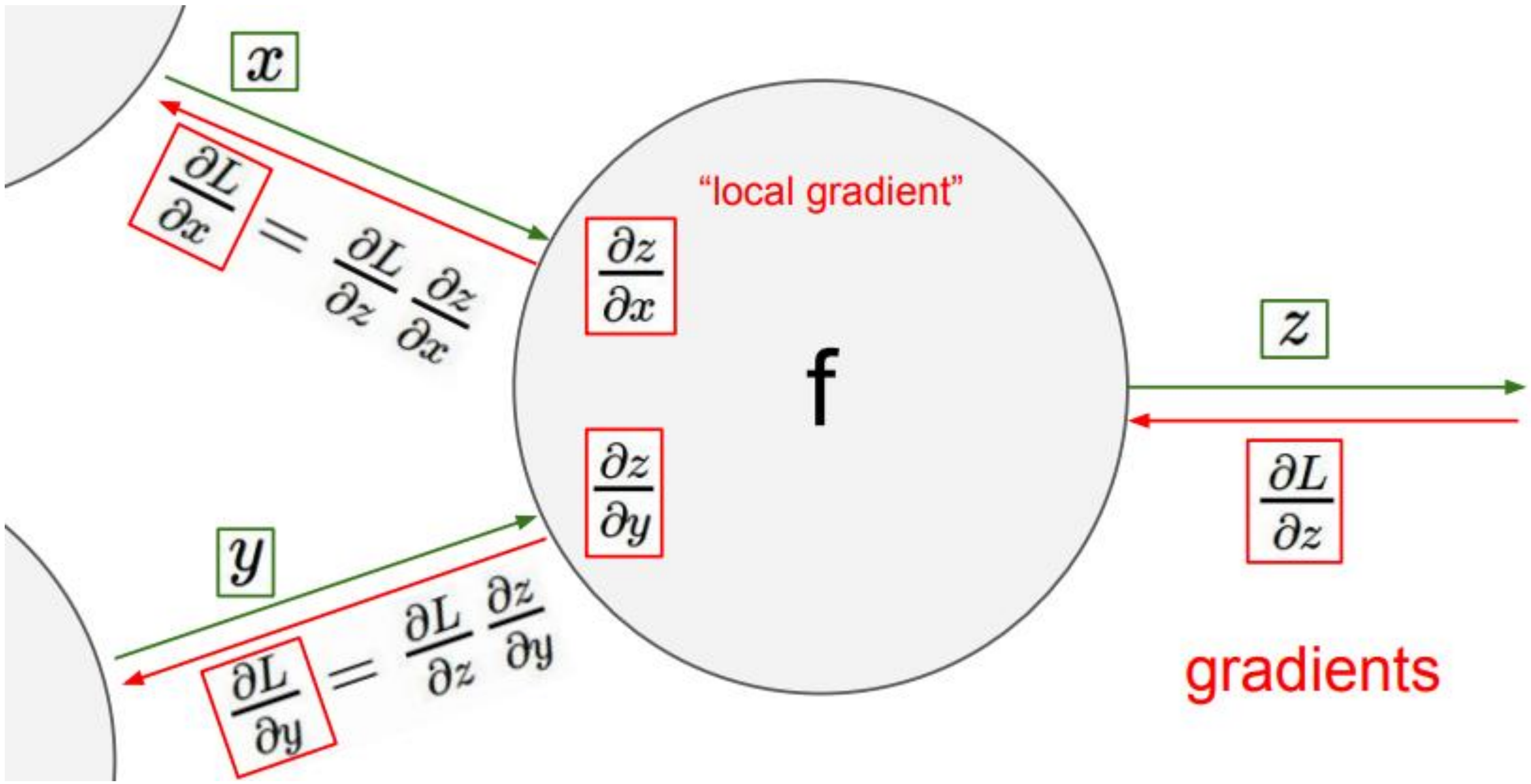
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

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

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



어떤 함수를 computational graph로 표현하고, input에 대한 gradient를 구하고 싶은데, 이는 최종 Loss로부터 각각의 local gradien를 **chain rule**를 적용해 뒤에서부터 계산하면 구할수 있다. 뒤에서부터 오기때문에 ‘**Backpropagation**’이라고 함

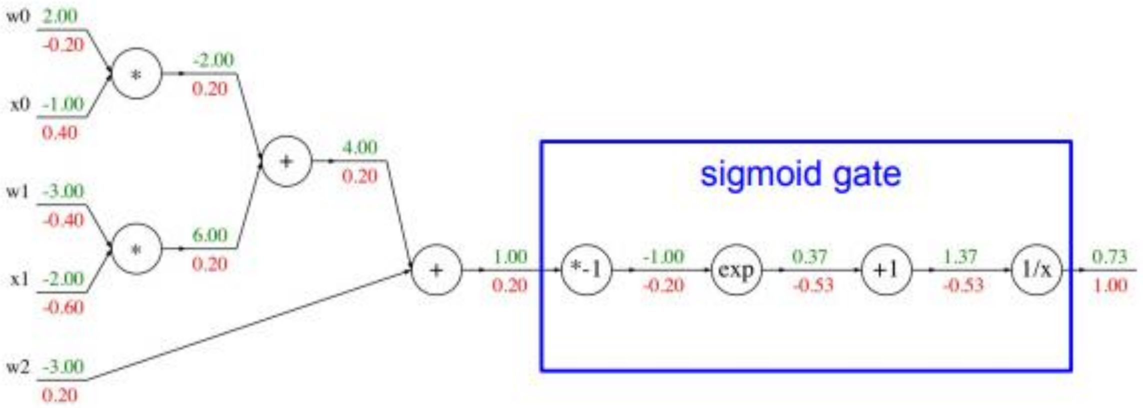


<Backpropagation의 과정>

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

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

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

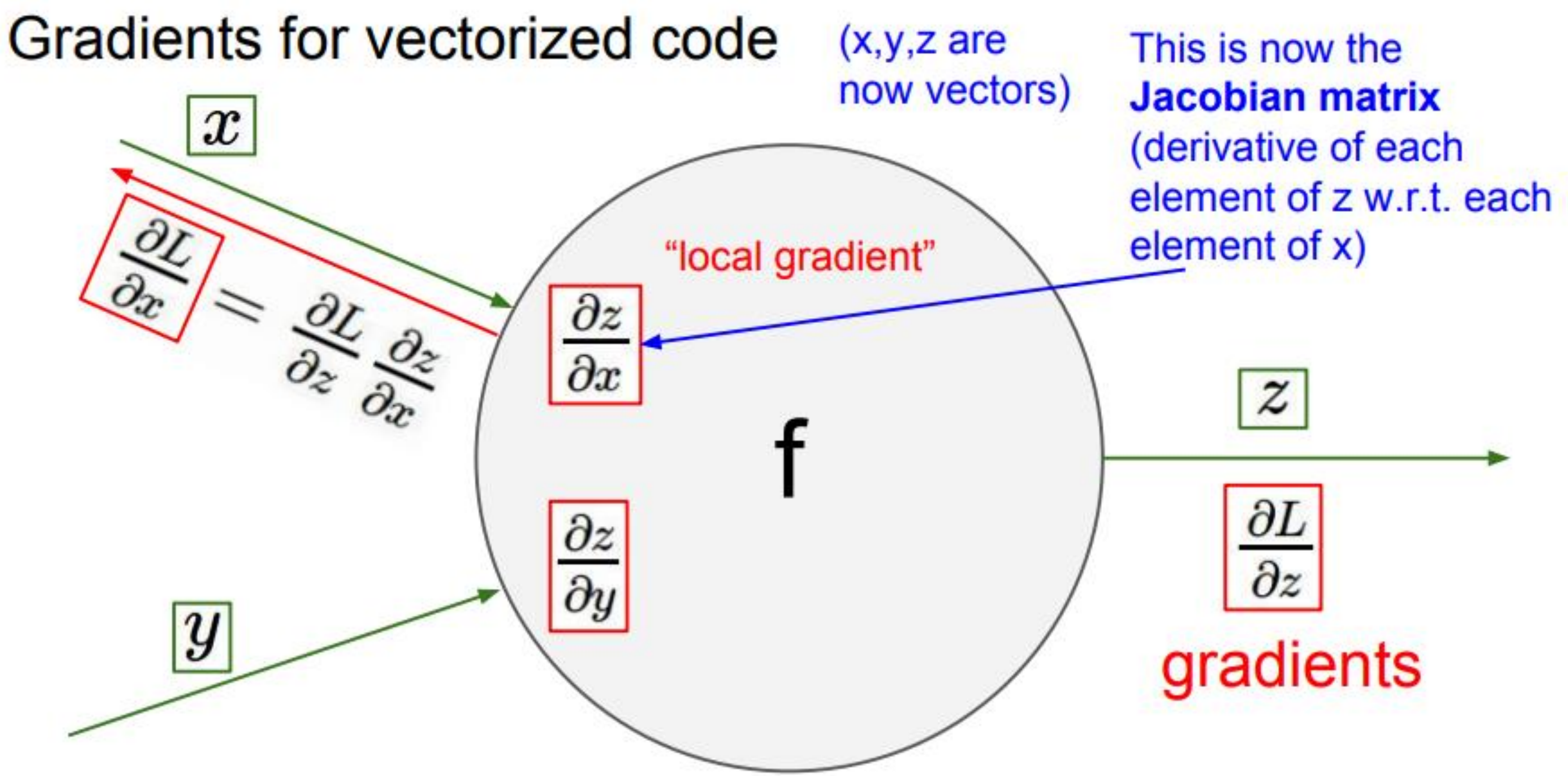


add gate: gradient distributor

max gate: gradient router

mul gate: gradient switcher

Sigmoid gate같이 미분값을 미리 계산할수 있는 부분이면 이 내부의 과정을 하나하나 계산하기 보다는 미리 계산된 식에 대입하면 빠르게 계산할수 있다.
Add gate는 local gradient가 1이기 때문에 뒤쪽의 gradient를 그대로 전달하는 역할을 하여 gradient distributor라 하고,
max gate는 값이 큰 한쪽의 값만을 뒤쪽으로 그대로 전달하여 gradient router,
mul gate는 뒤쪽값에서 input의 서로의 값을 바꿔서 곱하게 되어 gradient switcher라고 함



Gradient를 계산하는것은 하나의 값이 아니라 벡터에 대해서 계산하는 경우가 많다.
그에 따라 local gradient를 chain rule을 적용해서 계산하는 과정은 **jacobian matrix**로 표현된다.

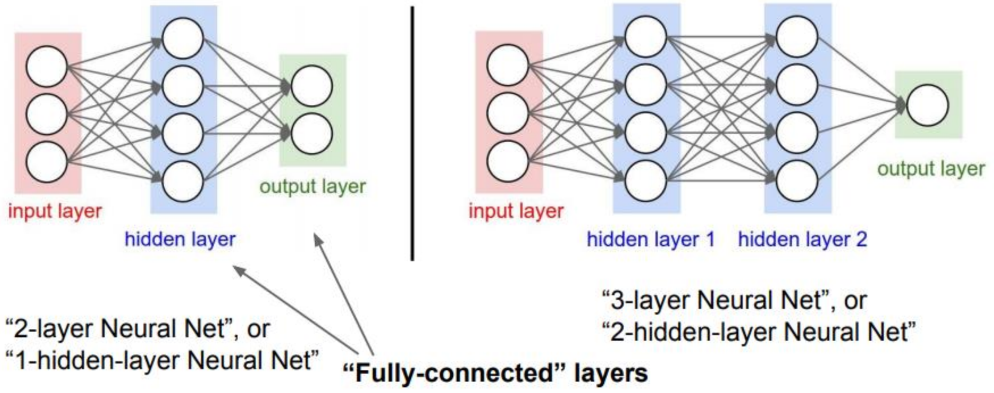
Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Neural networks: Architectures



Input->여러 개의 hidden layer ->output layer 구조의 **n-layer Neural Network**
:서로 모든 노드에 관여하여 값을 도출하여 Fully Connected 되어있다.