Backprop and Neural Networks

Named Entity Recognition (NER)

↳ find and classify names in text

ex. Samuel Quinn was arrested in the Hilton Hotel in Paris in April 1989.
PER PER LOC LOC LOC DATE DATE

How? build up a context window of neighboring words
hand-labeled data로 logistic classifier 훈련

$$J_t(\theta) = \sigma(s) = \frac{1}{1+e^{-s}}$$

predicted model probability of class

word vector를 neural net에 넣고 행렬곱, 확률로 변환

Computing gradient by hand

Matrix calculus : fully vectorized gradients

Gradients

- one input & one output

$$f(x) = x^3$$

- It's gradient is its derivative

$$\frac{df}{dx} = 3x^2$$

- "input을 조금 바꾸면 output은 얼마나 바뀔까?"

- n inputs & one output

$$f(x) = f(x_1, x_2, \cdots, x_n)$$

- It's gradient is a [vector of partial derivatives] with respect to each output

$$\frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right]$$

- n inputs & m outputs

$$f(x) = [f_1(x_1, x_2, \cdots, x_n), \cdots f_m(x_1, x_2, \cdots, x_n)]$$

- It's Jacobian is an $m \times n$ matrix of partial derivatives

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \left(\frac{\partial f}{\partial x}\right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

$\Rightarrow$ Have every possible partial derivative of an output variable

Chain Rule

- For composition of one-variable functions: multiply derivatives

$$z = 3y$$
$$y = x^2$$
$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = 3 \cdot 2x = 6x$$

- For multiple variables at once: multiply Jacobians

$$h = f(z)$$
$$z = Wx + b$$
$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial z}\frac{\partial z}{\partial x} = \cdots$$

(1) $\frac{\partial}{\partial x}(Wx+b) = W$, (2) $\frac{\partial}{\partial b} = I$, (3) $\frac{\partial}{\partial u}(u^T h) = h^T$

예시.

$$s = u^T h$$
$$h = f(Wx + b)$$
$$x$$

} 일 때, $\frac{\partial s}{\partial b}$ 찾기

1. Break up equations into simple pieces

$$s = u^T h$$
$$h = f(Wx + b) \Rightarrow h = f(z)$$
$$z = Wx + b$$
$$x$$

2. Apply the chain rule

$$\frac{\partial s}{\partial b} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial b}$$

$$= u^T \cdot \mathrm{diag}(f'(z)) \cdot I = u^T \circ f'(z)$$

Hadamard product

δ

Re-using computation

$$\frac{\partial s}{\partial W} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial W}$$

$$= u^T \cdot \mathrm{diag}(f'(z)) \cdot \frac{\partial z}{\partial W}$$

δ

$$= \delta^T x^T$$

local error signal@z / local input signal

※ Why transpose?

$$\frac{\partial s}{\partial W} = \delta^T x^T$$

$[n \times m] \quad [n \times 1] \quad [1 \times m]$

this makes the dimensions work out!

Backpropagation 역전파

re-use derivatives computed for higher layers in computing derivatives for lower layers to minimize computation

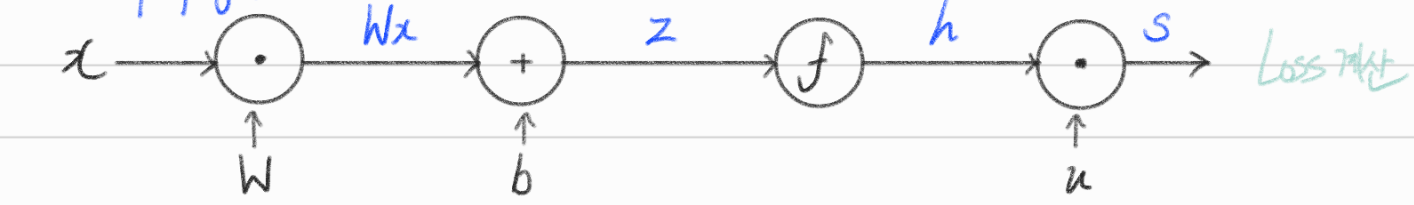
input: x

$$z = Wx + b$$

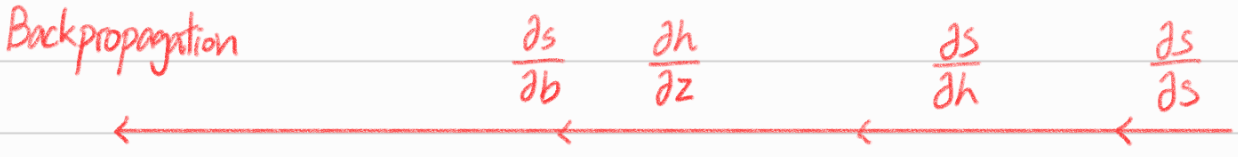
$$h = f(z) \quad \text{(ex. Sigmoid)}$$

$$s = u^T h$$

Forward propagation



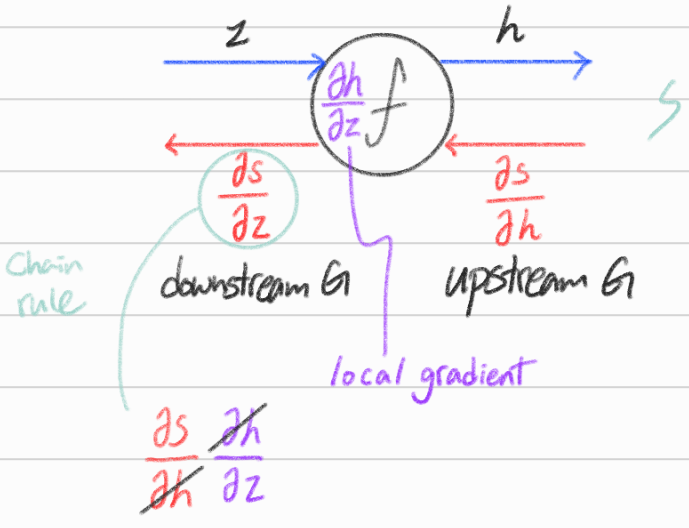
Backpropagation



- pass along gradients

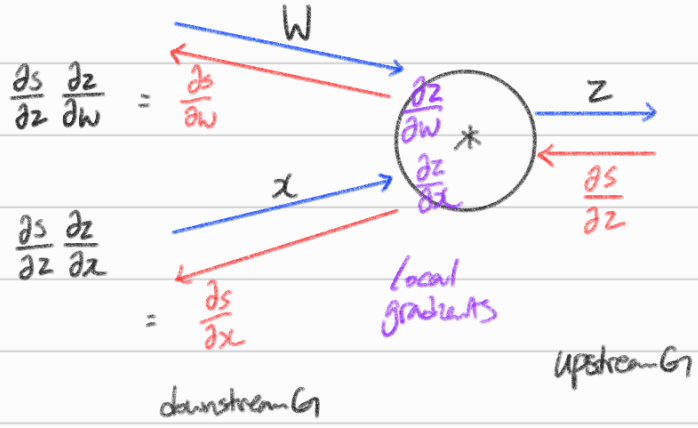
Single Node

- Node receives an "upstream gradient"
- Goal is to pass on the correct "downstream gradient"

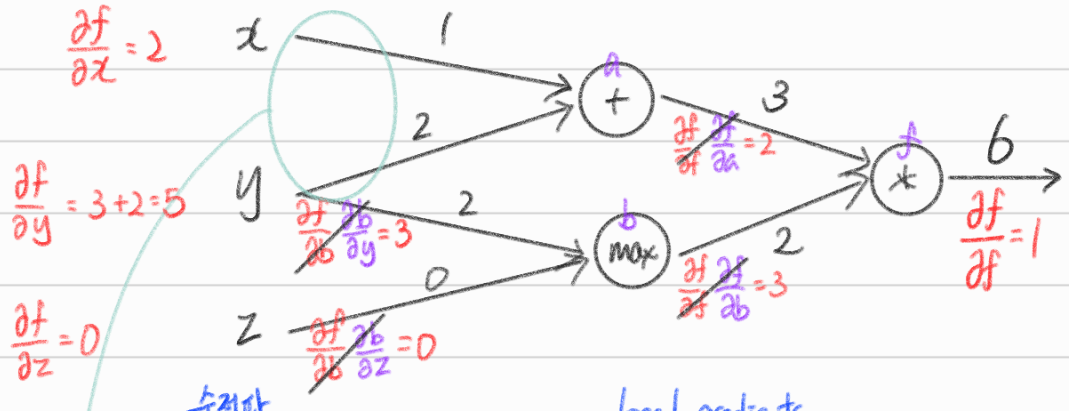


즉, downstream gradient = upstream gradient × local gradient

Multiple inputs



ex. $f(x,y,z) = (x+y)\max(y,z)$ $\quad x=1, y=2, z=0$



순전파

$a = x+y$

$b = \max(y,z)$

$f = ab$

local gradients

$\frac{\partial a}{\partial x} = 1, \quad \frac{\partial a}{\partial y} = 1$

$\frac{\partial b}{\partial y} = 1(y>z) = 1, \quad \frac{\partial b}{\partial z} = 1(z>y) = 0$

$\frac{\partial f}{\partial a} = b = 2, \quad \frac{\partial f}{\partial b} = a = 3$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x} = 2 \times 1 = 2$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial y} = 2 \times 1 = 2$$

Downstream = Upstream × Local $\qquad D = U \cdot L$

Node Intuitions

+ : distributes the upstream gradient

max : routes "

* : switches "

계산: Compute all the gradients at once

Analogous to using δ when we computed gradients by hand

Fprop : visit nodes in topological sort order

Bprop : - initialize output gradient
- visit nodes in reverse order

big O() complexity of fprop & bprop is the same