FVE3011

Backpropagation, activation function

담당교수: 최 학남 (xncui@inha.ac.kr)

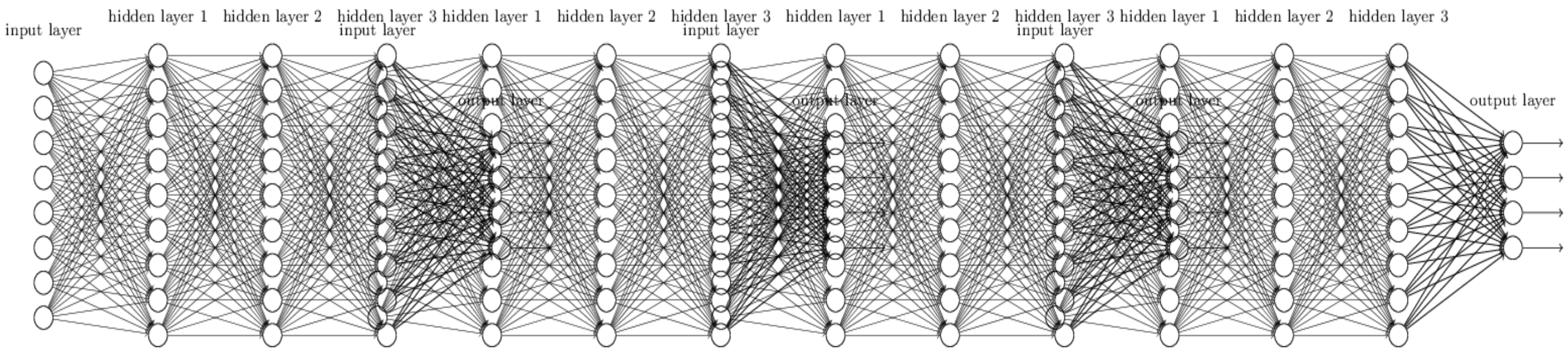
인하대학교

Contents

- Backpropagation

- Activation function

How to train?



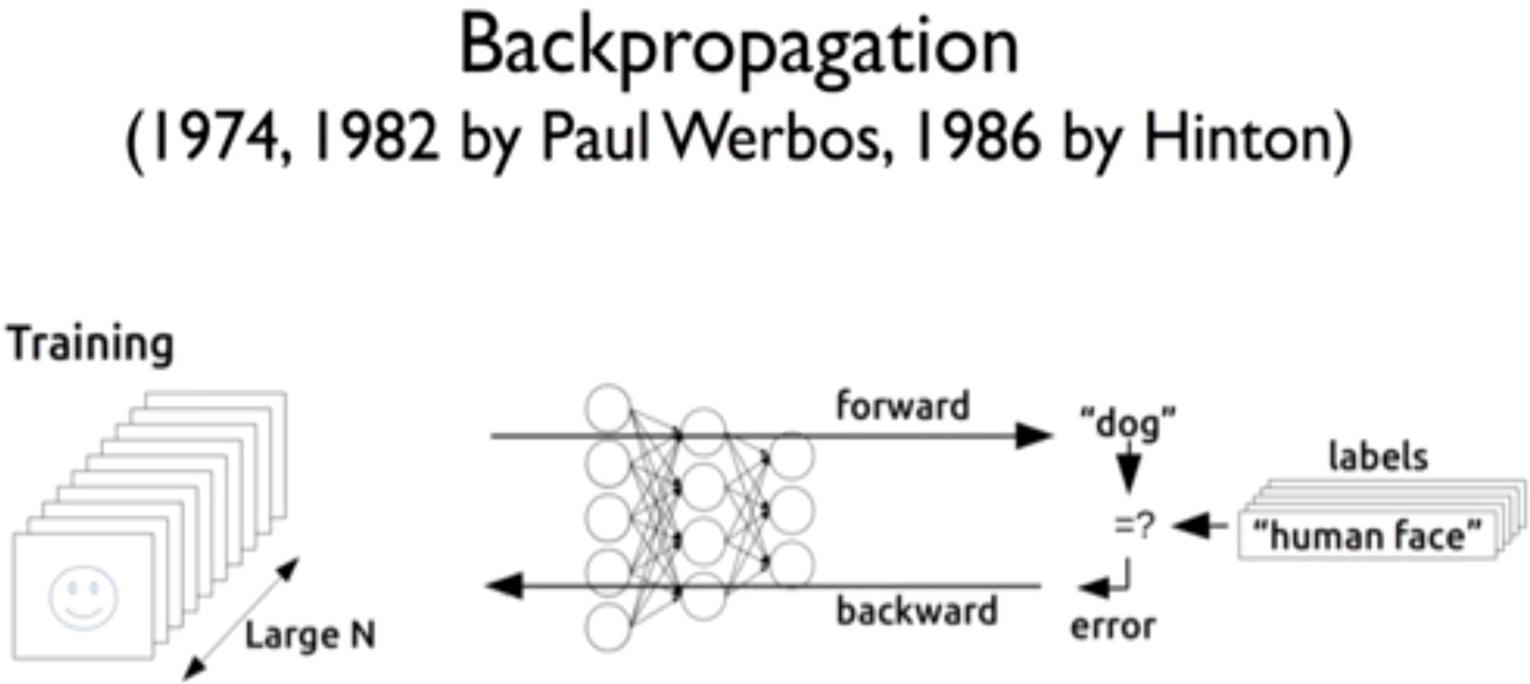
- GDA(Gradient descent algorithm)

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)}) x^{(i)}$$

$$w = w - \alpha \frac{\partial E}{\partial w}$$

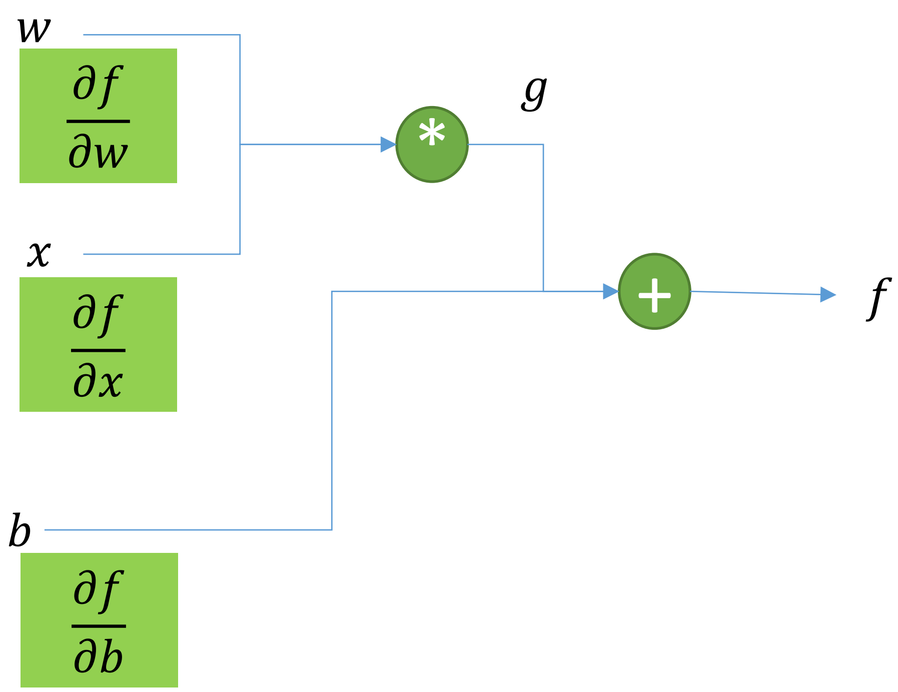
Backpropagation

- Solve the XOR problem using backpropagation



Back propagation (chain rule)

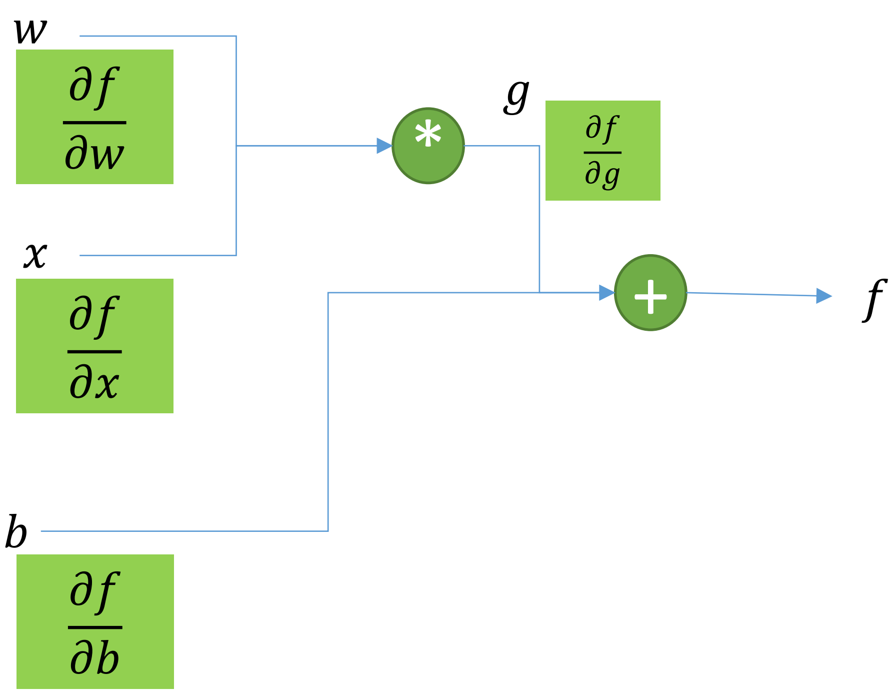
$f = wx + b$ $\quad\quad$ $g = wx$ $\quad\quad$ $f = g + b$



Back propagation (chain rule)

$\frac{\partial g}{\partial w} = x$ $\frac{\partial g}{\partial x} = w$ $\frac{\partial f}{\partial g} = 1$ $\frac{\partial f}{\partial b} = 1$

$f = wx + b$ $g = wx$ $f = g + b$



(1) Forward : w=-2,x=5,b=3

(2) backward :

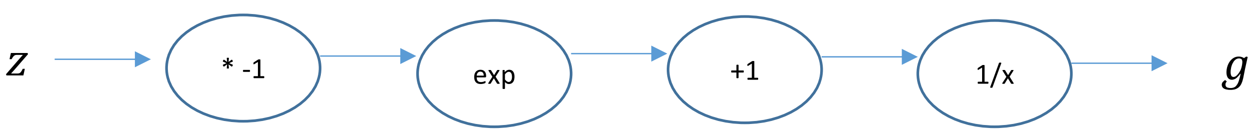
$$f(g(x)) = y$$

$$\frac{\partial y}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Sigmoid

Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$

$$\boxed{\frac{\partial g(z)}{\partial z} = ?}$$



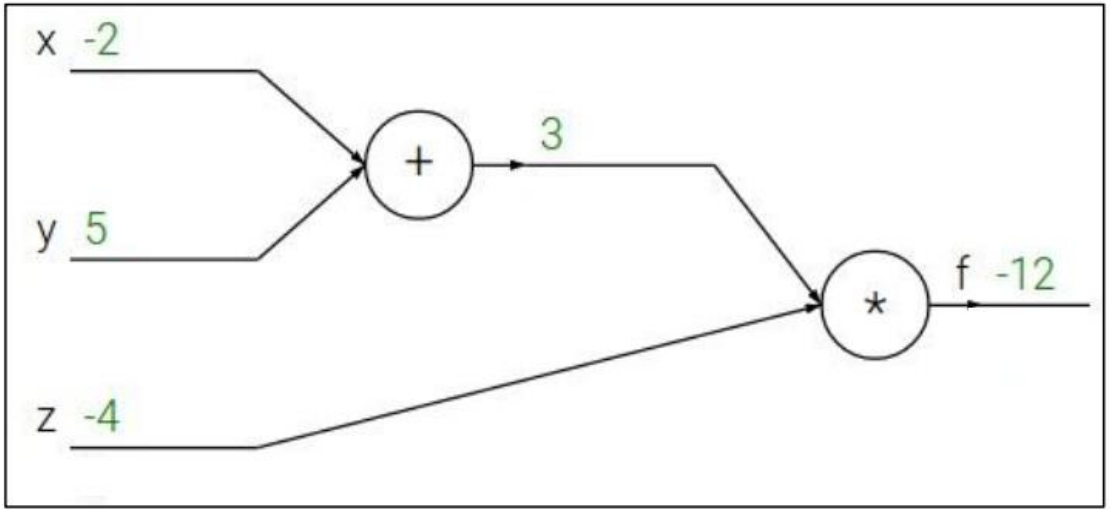
$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x}\sigma(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{1+e^{-x}}\right) \\
&= \frac{\mathrm{d}}{\mathrm{d}x}\left(1+e^{-x}\right)^{-1} \\
&= -1 * (1+e^{-x})^{-1-1} * -1 * (e^{-x}) \\
&= -1 * (1+e^{-x})^{-2} * -(e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} * \frac{e^{-x}}{(1+e^{-x})} \\
&= \frac{1}{(1+e^{-x})} * \frac{1+e^{-x}-1}{(1+e^{-x})} \\
&= \frac{1}{(1+e^{-x})} * \left(\frac{(1+e^{-x})}{(1+e^{-x})} - \frac{1}{(1+e^{-x})}\right) \\
&= \frac{1}{(1+e^{-x})} * \left(1 - \frac{1}{(1+e^{-x})}\right)
\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{dx}}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Start from Simple

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

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

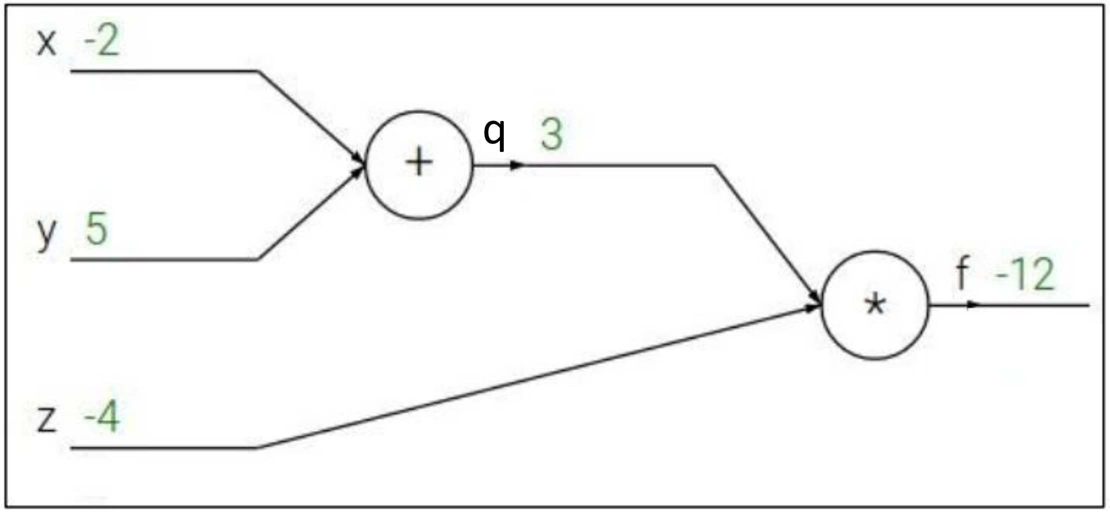


Slide from CS231n

Start from Simple

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

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



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

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

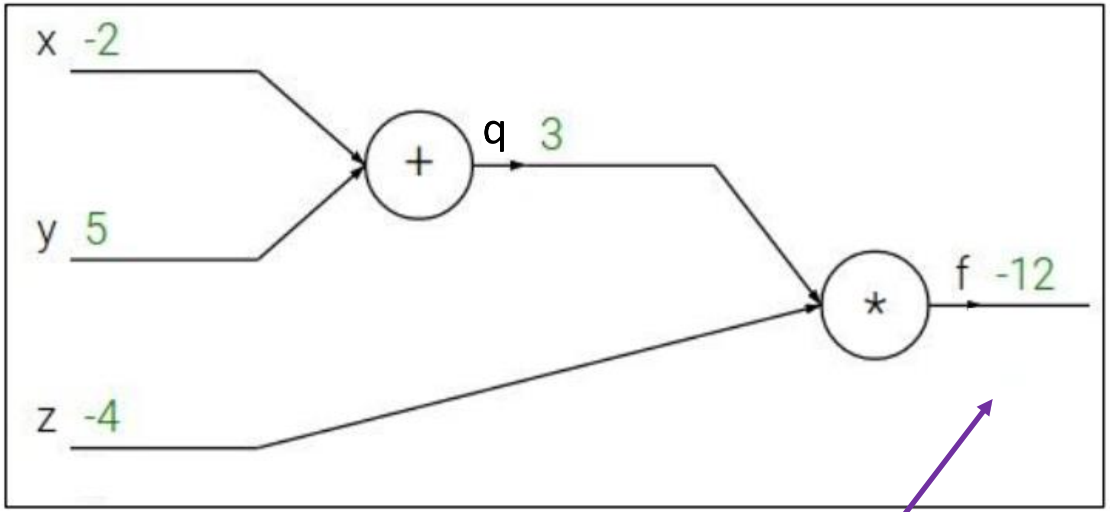
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Slide from CS231n

Start from Simple

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

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



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

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

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

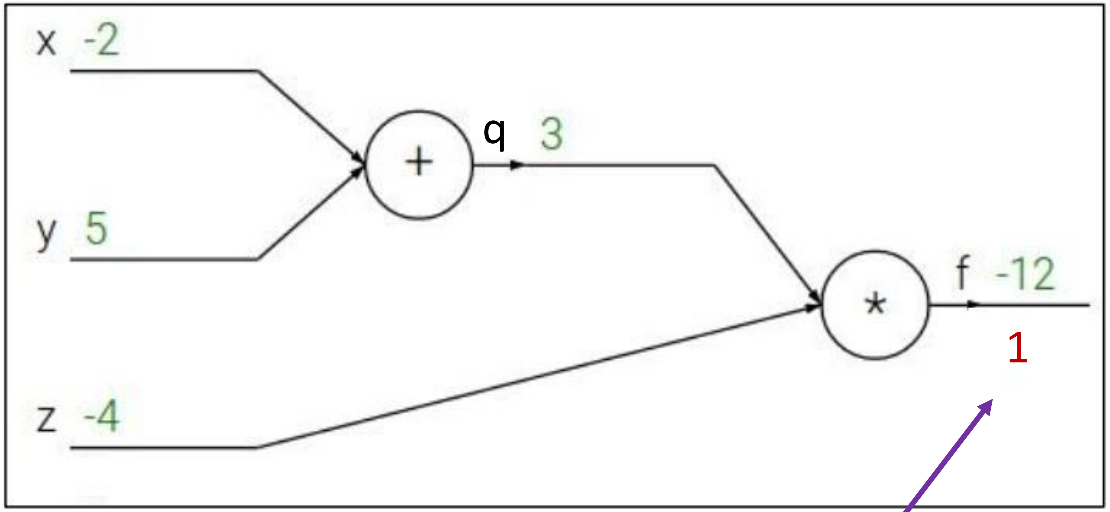
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Slide from CS231n

Start from Simple

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

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



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

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

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

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

Slide from CS231n

Start from Simple

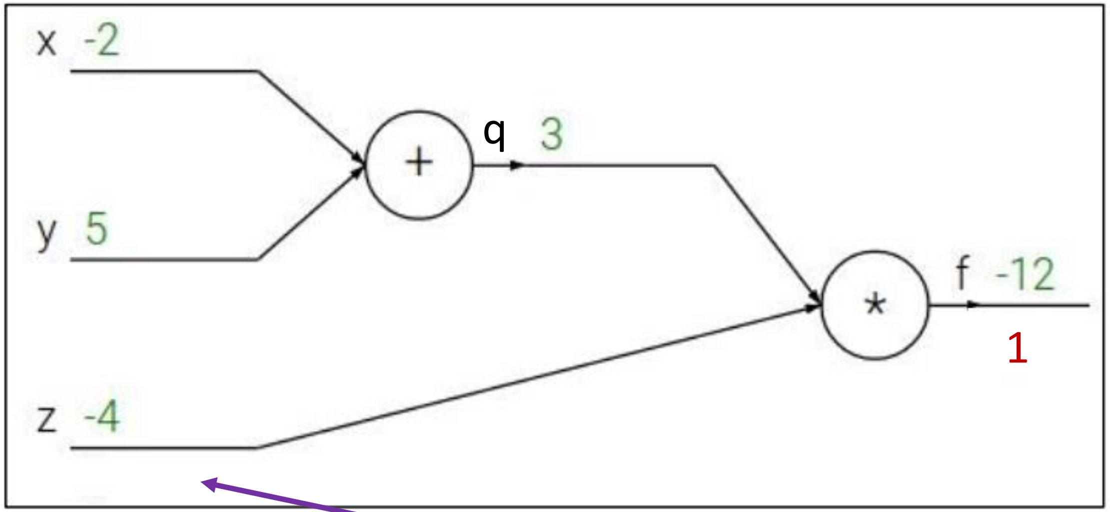
$f(x, y, z) = (x + y)z$

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



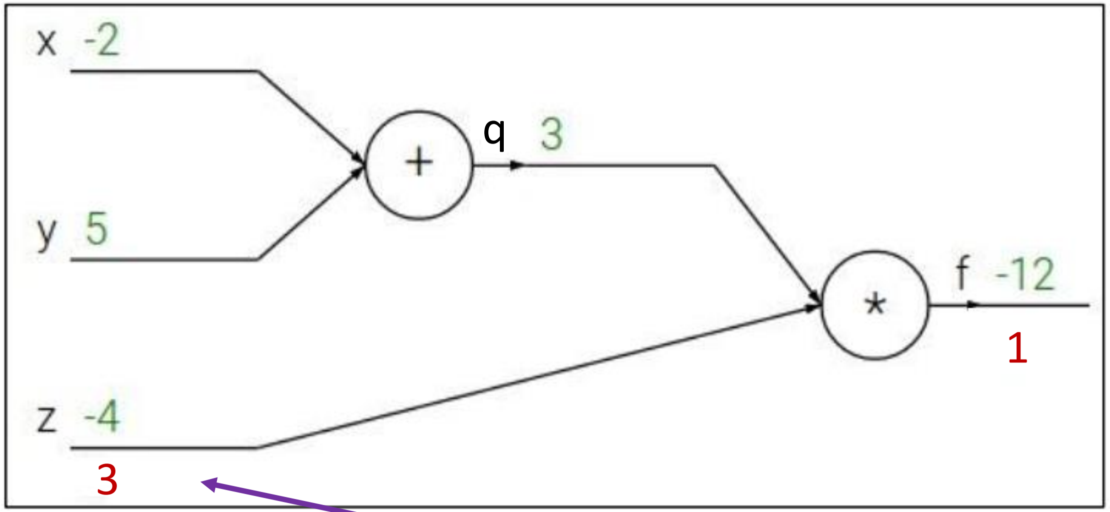
$\frac{\partial f}{\partial z}$

Slide from CS231n

Start from Simple

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

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



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

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

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

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

Slide from CS231n

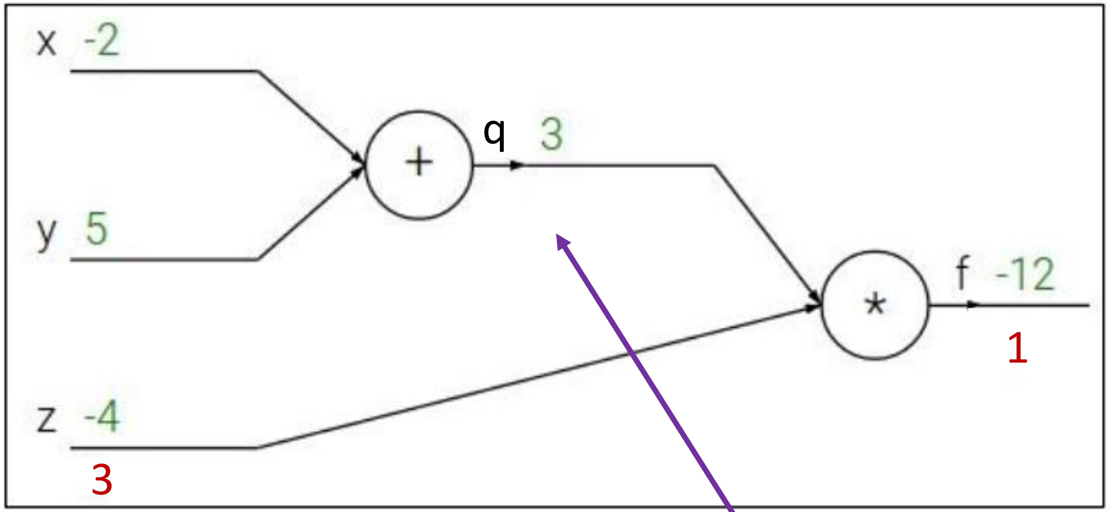
Start from Simple

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

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

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

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



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

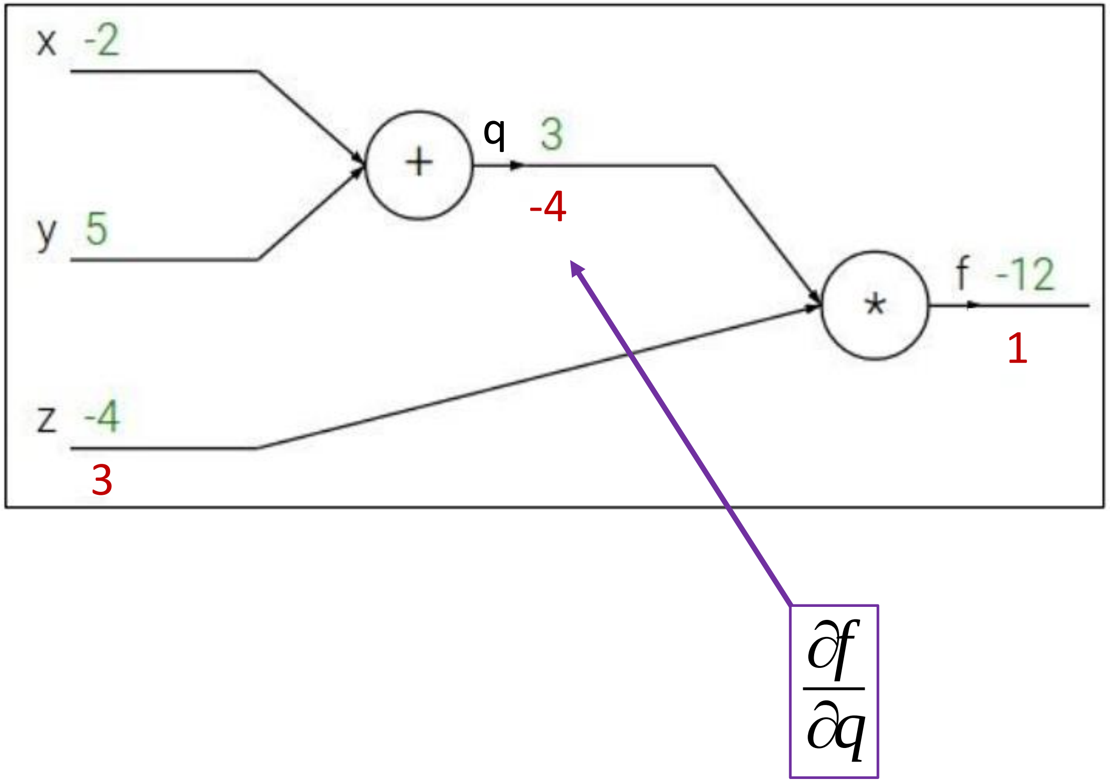
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Slide from CS231n

Start from Simple

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

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

Slide from CS231n

Start from Simple

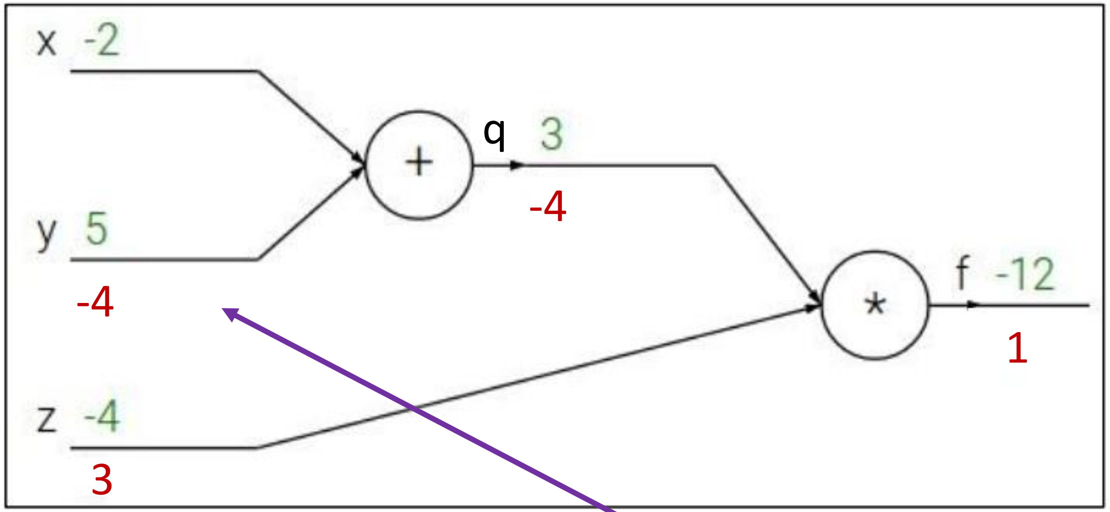
$f(x, y, z) = (x + y)z$

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:

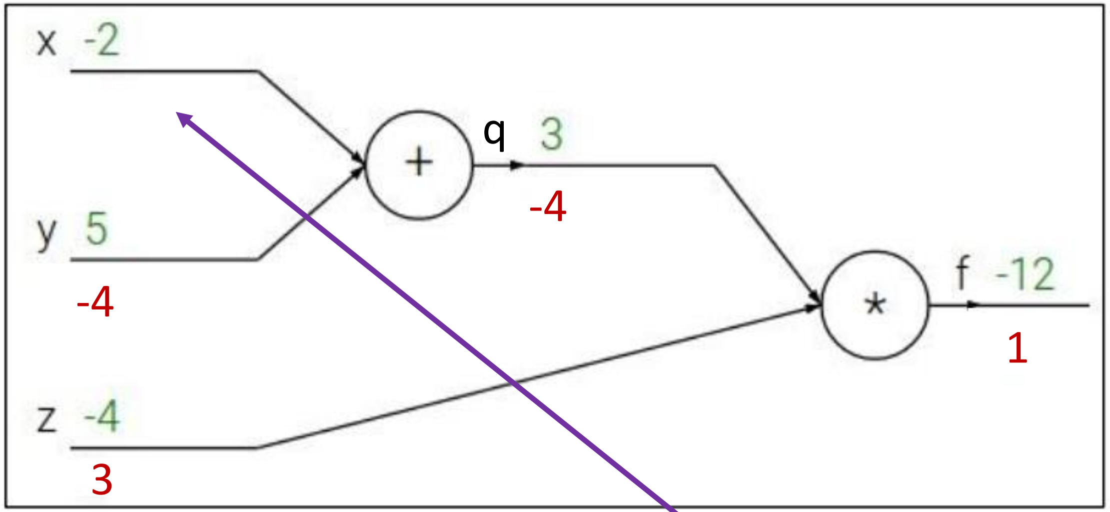
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

Slide from CS231n

Start from Simple

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

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



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

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

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

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

Slide from CS231n

Start from Simple

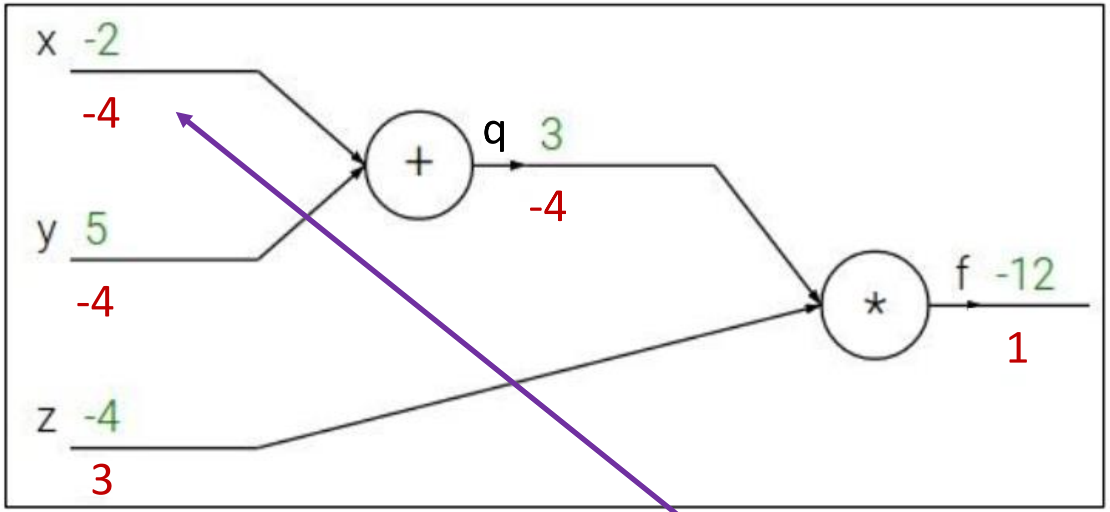
$f(x, y, z) = (x + y)z$

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



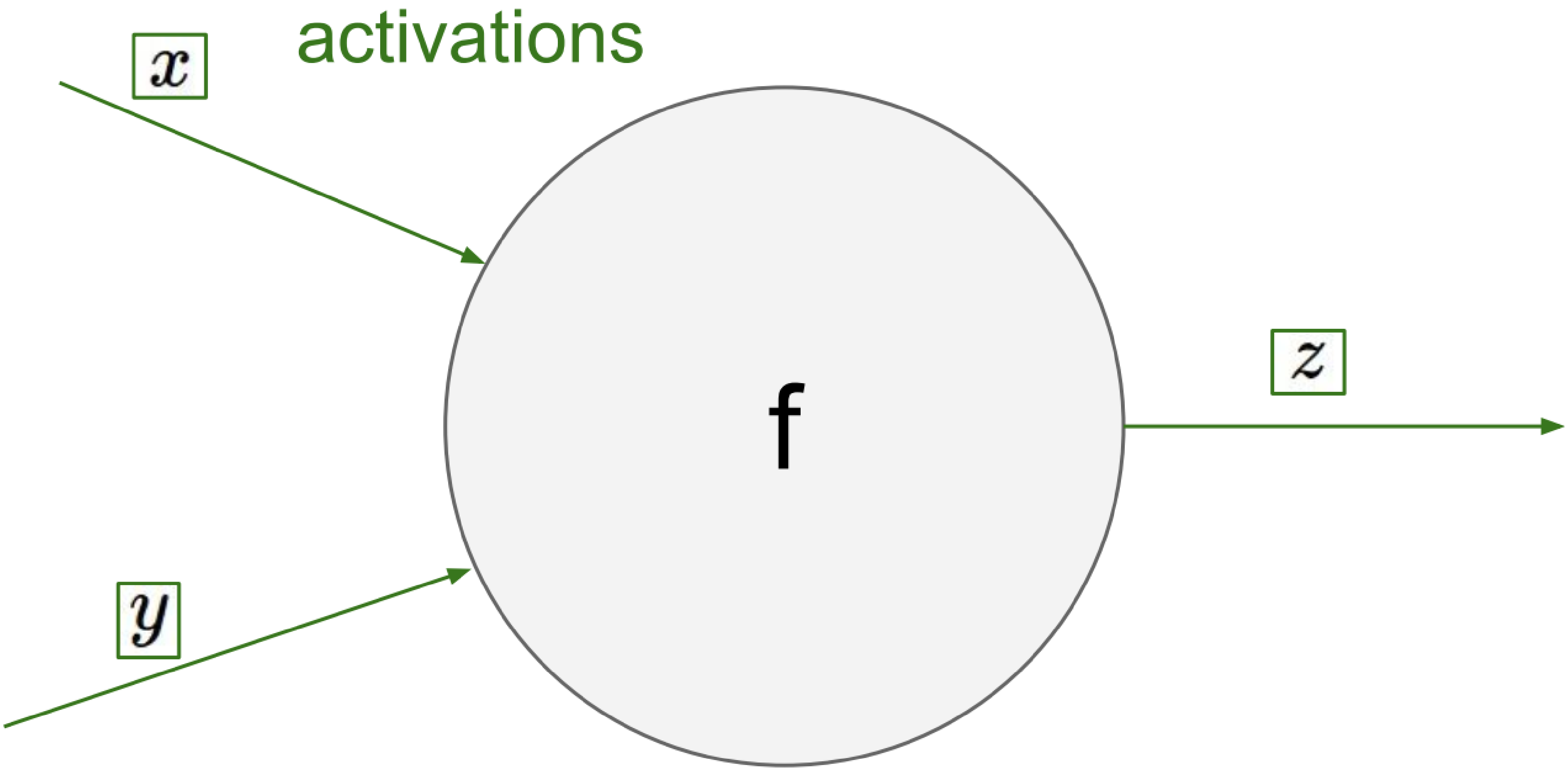
$$\frac{\partial f}{\partial x}$$

Chain rule:

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

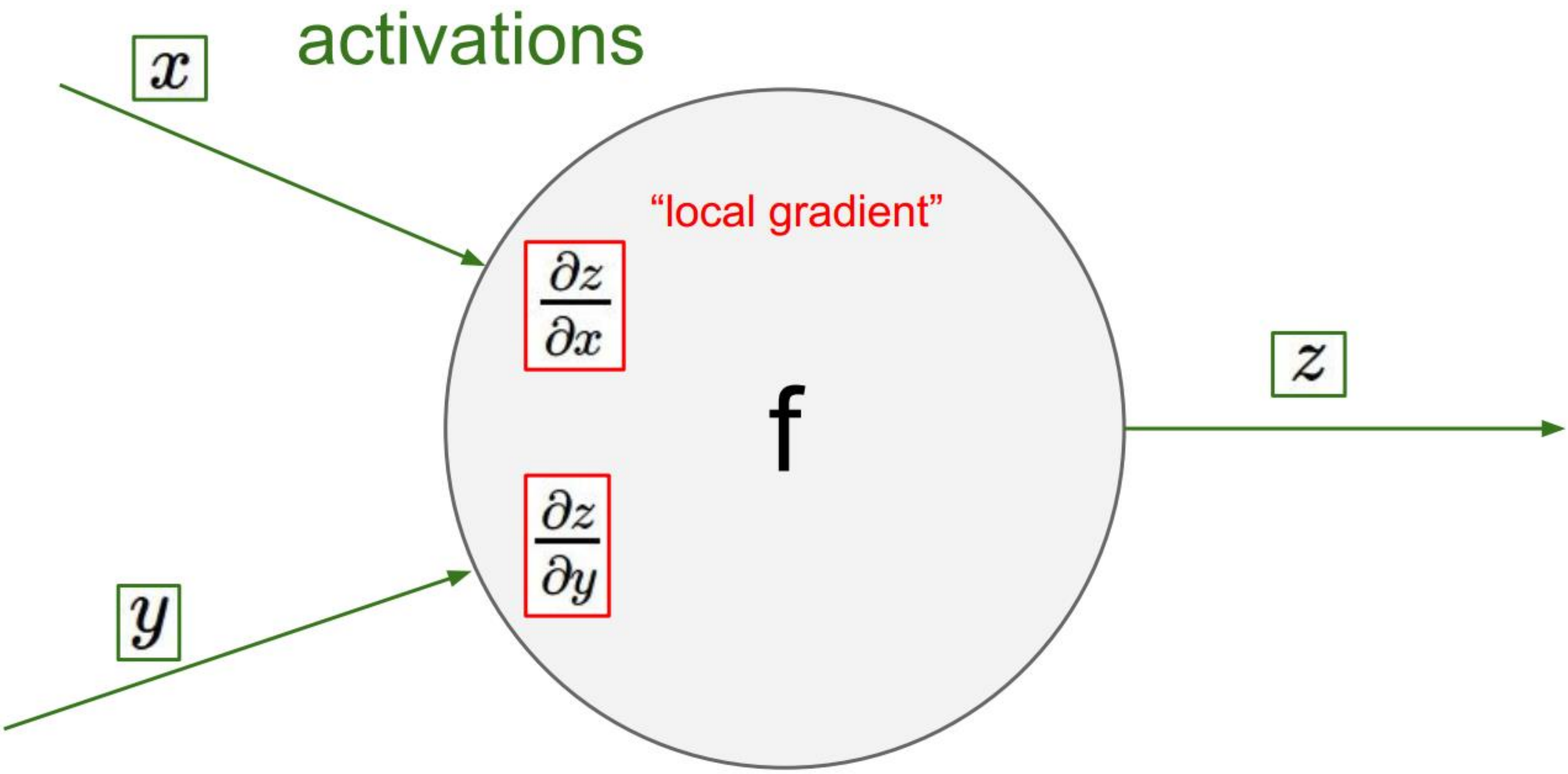
Slide from CS231n

Backpropagation



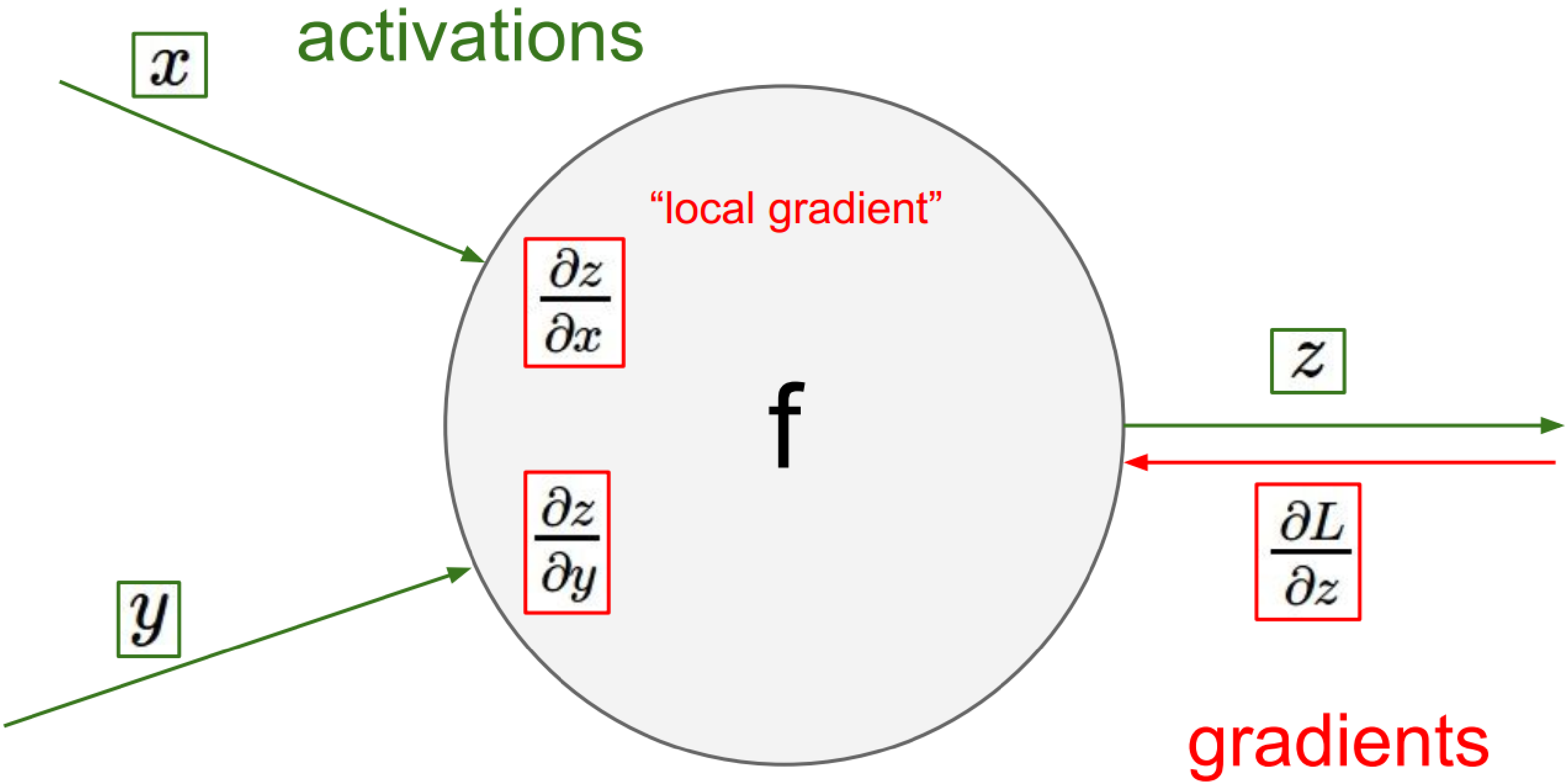
Slide from CS231n

Backpropagation



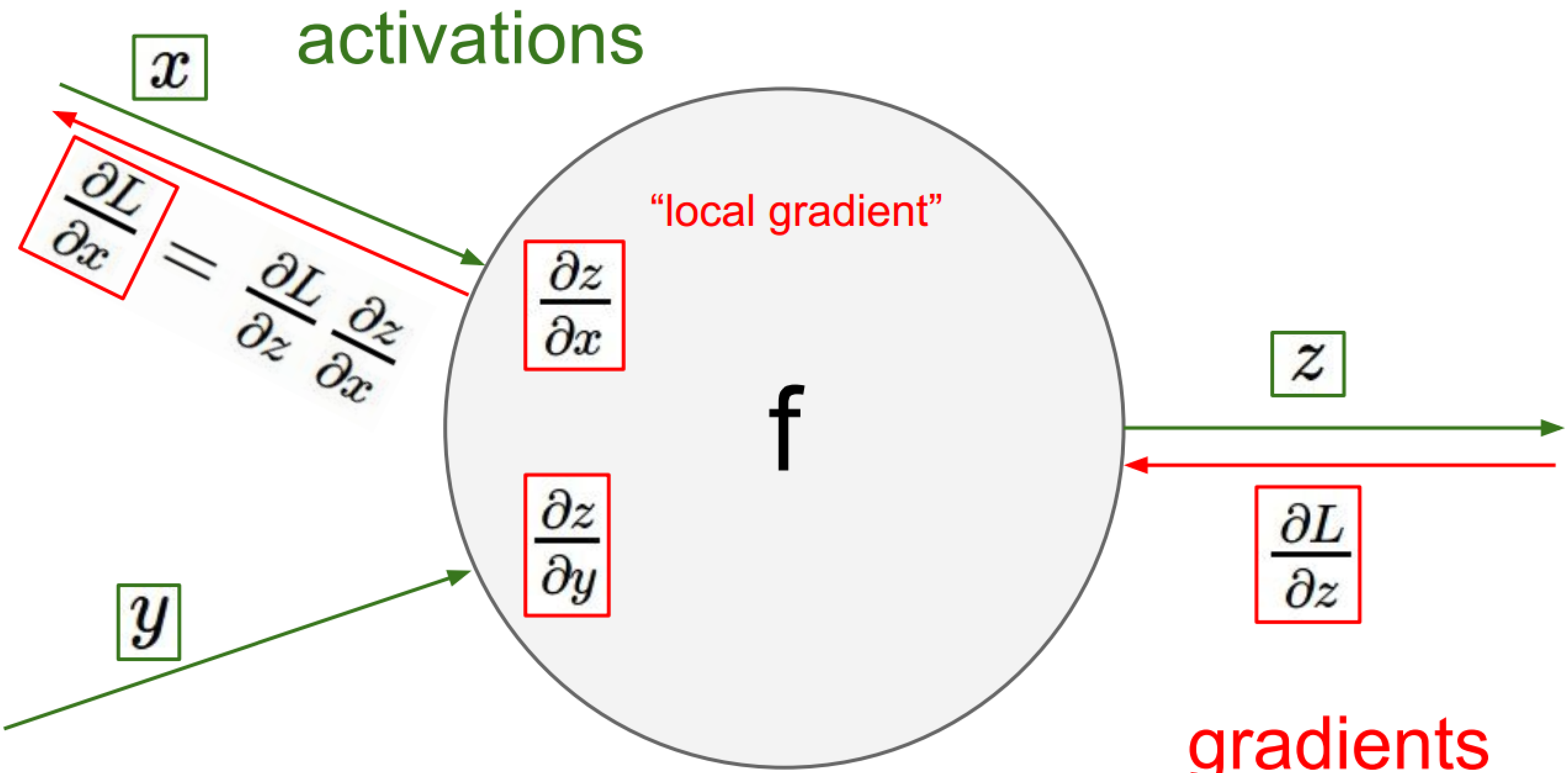
Slide from CS231n

Backpropagation



Slide from CS231n

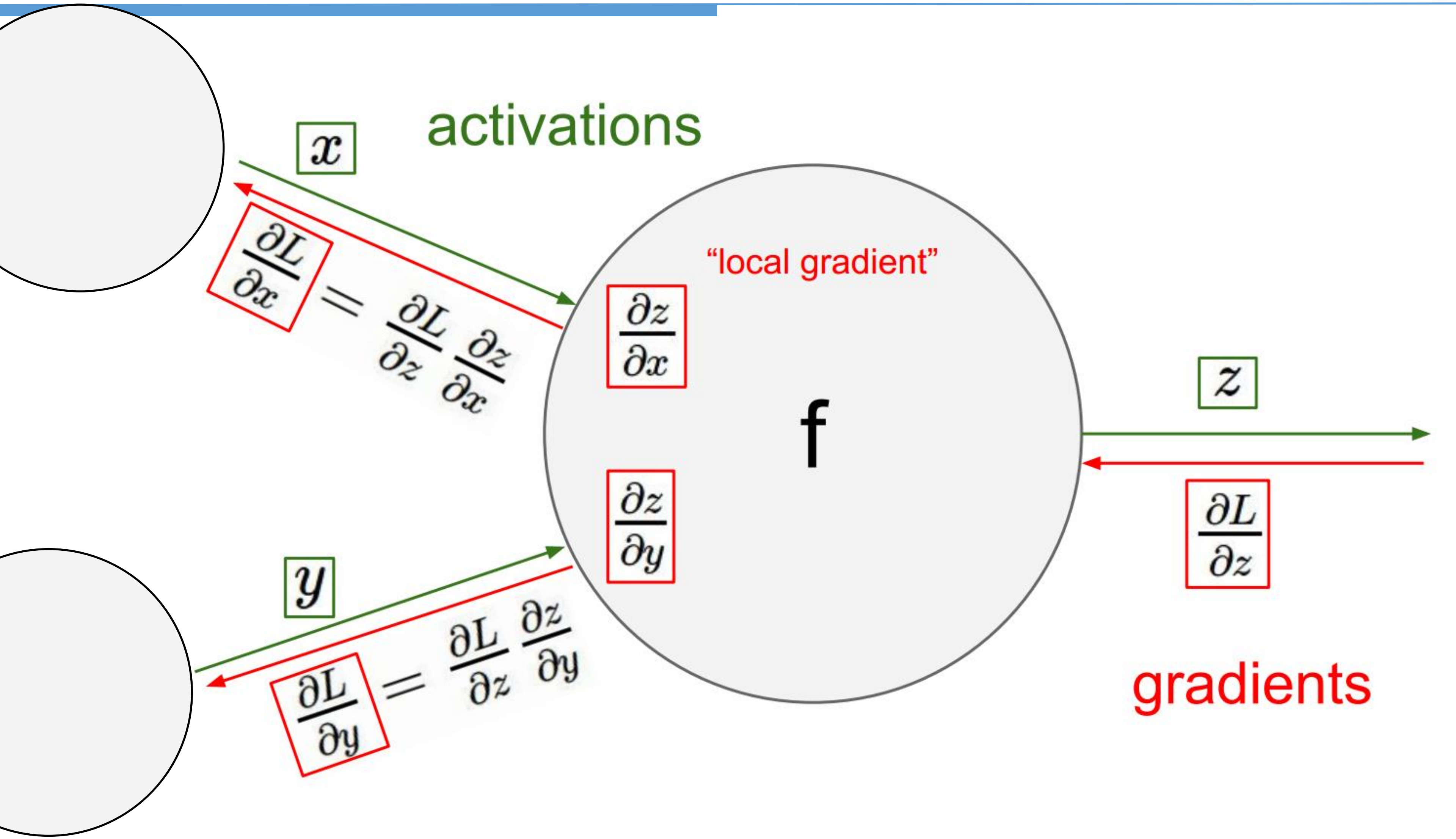
Backpropagation



Chain Rule

Slide from CS231n

Backpropagation

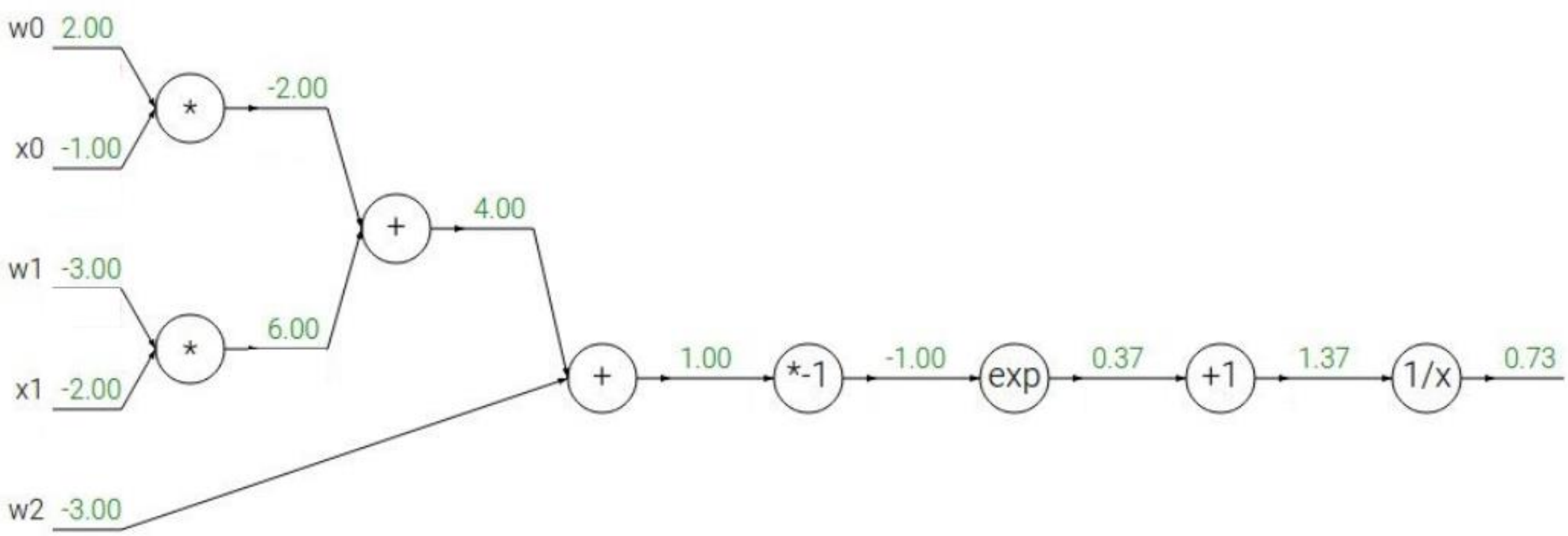


Chain rule

Slide from CS231n

Another Example

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

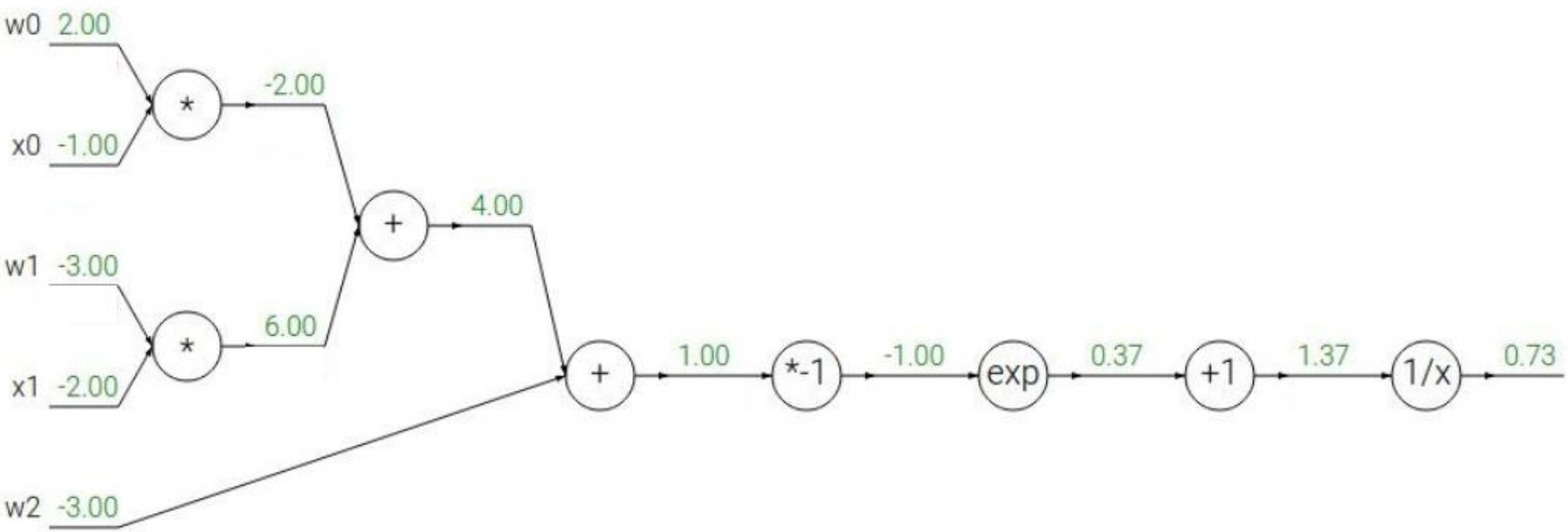


Computation graph of sigmoid neuron

Slide from CS231n

Another Example

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

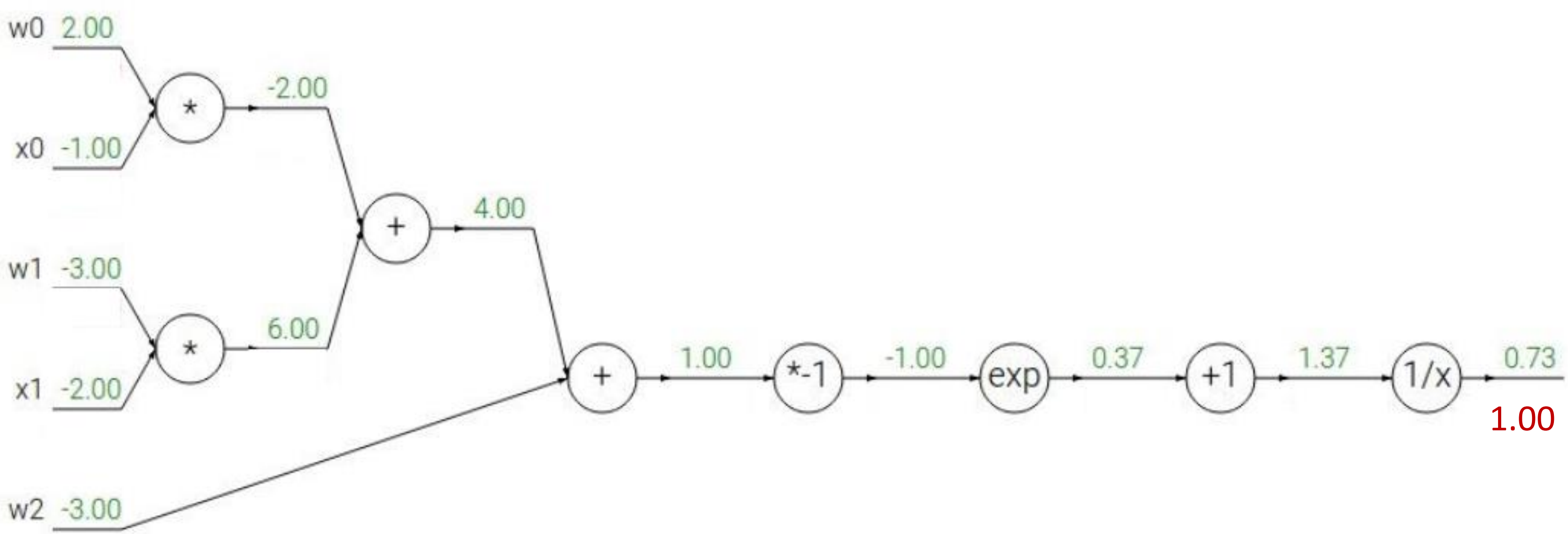


$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$	$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$
$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$	$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

Slide from CS231n

Another Example

Sigmoid Neuron: $$f(w,x) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



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

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

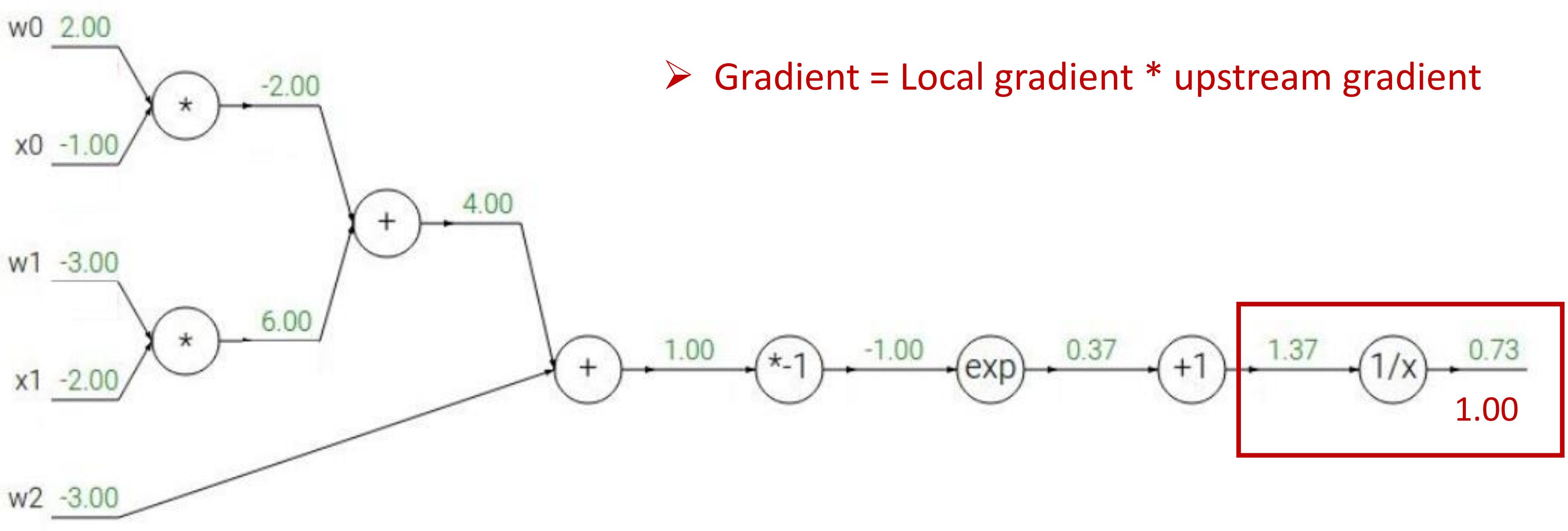
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

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

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

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

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

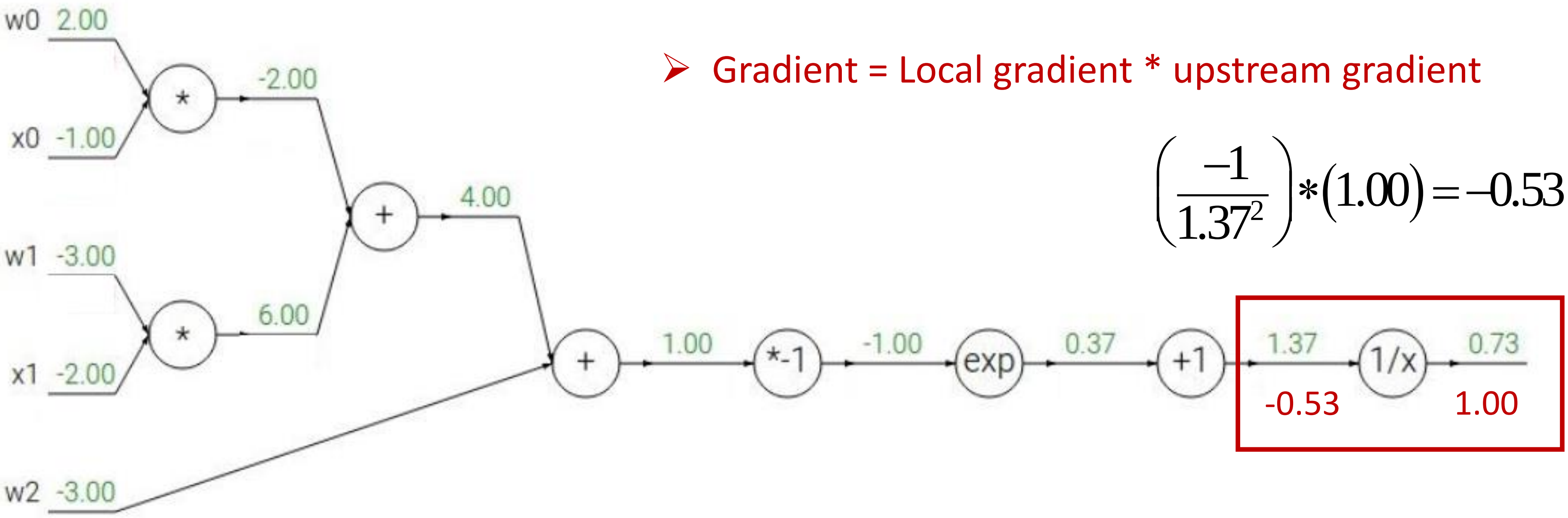
Slide from CS231n

Another Example

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

- Gradient = Local gradient * upstream gradient

$$\left(\frac{-1}{1.37^2}\right)*(1.00)=-0.53$$



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

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

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

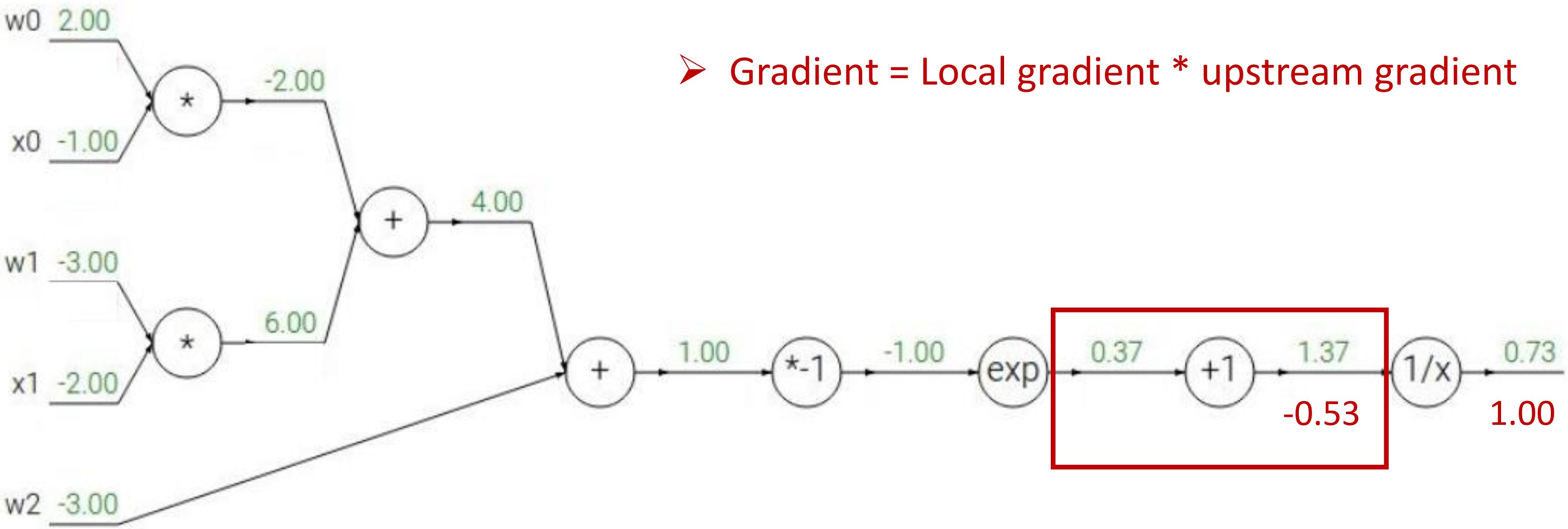
$$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$$

Slide from CS231n

Another Example

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

- Gradient = Local gradient * upstream gradient



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

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

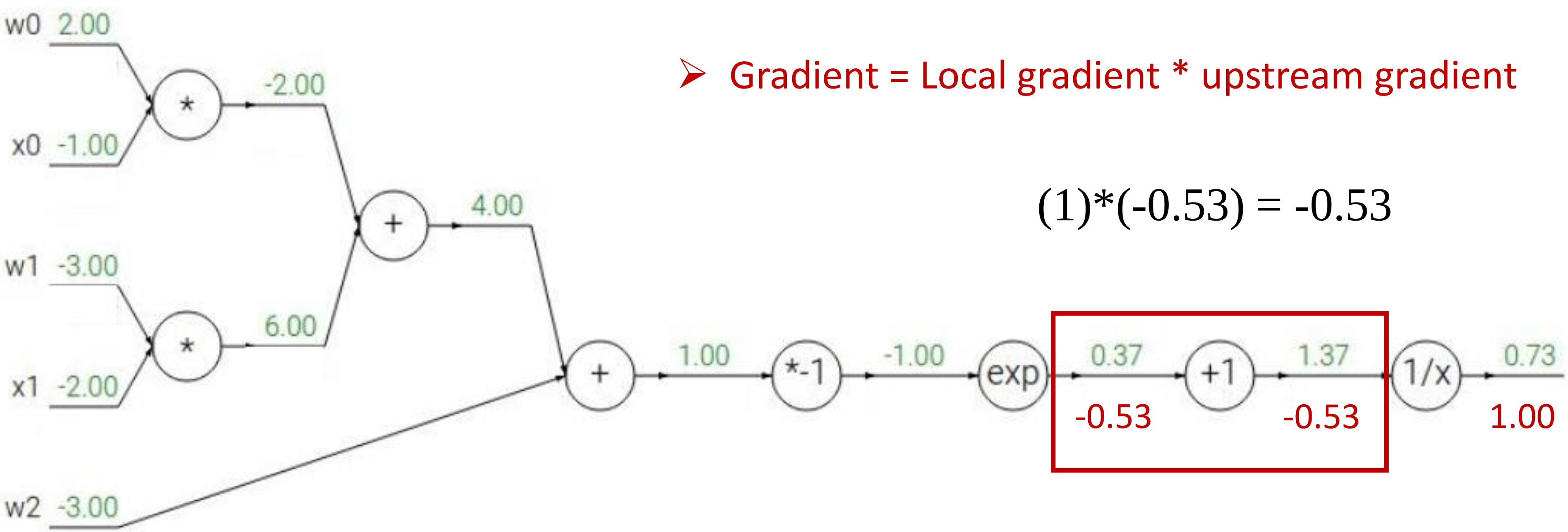
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

(1)*(-0.53) = -0.53

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

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

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

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

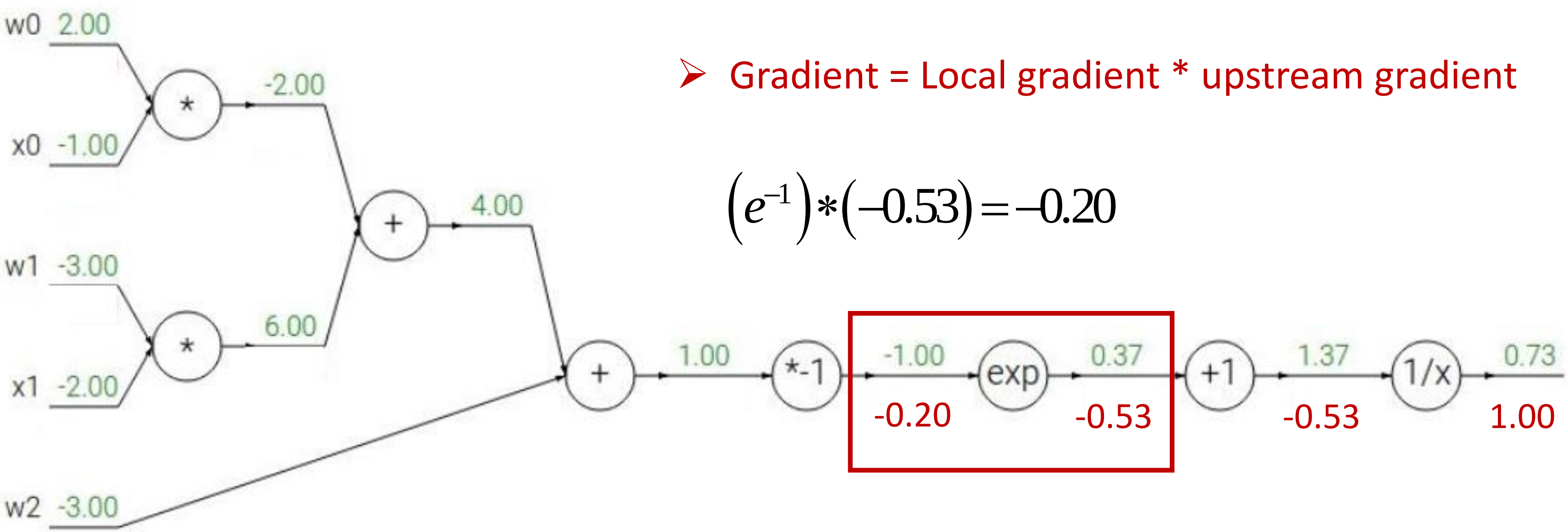
Slide from CS231n

Another Example

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

- Gradient = Local gradient * upstream gradient

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



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

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

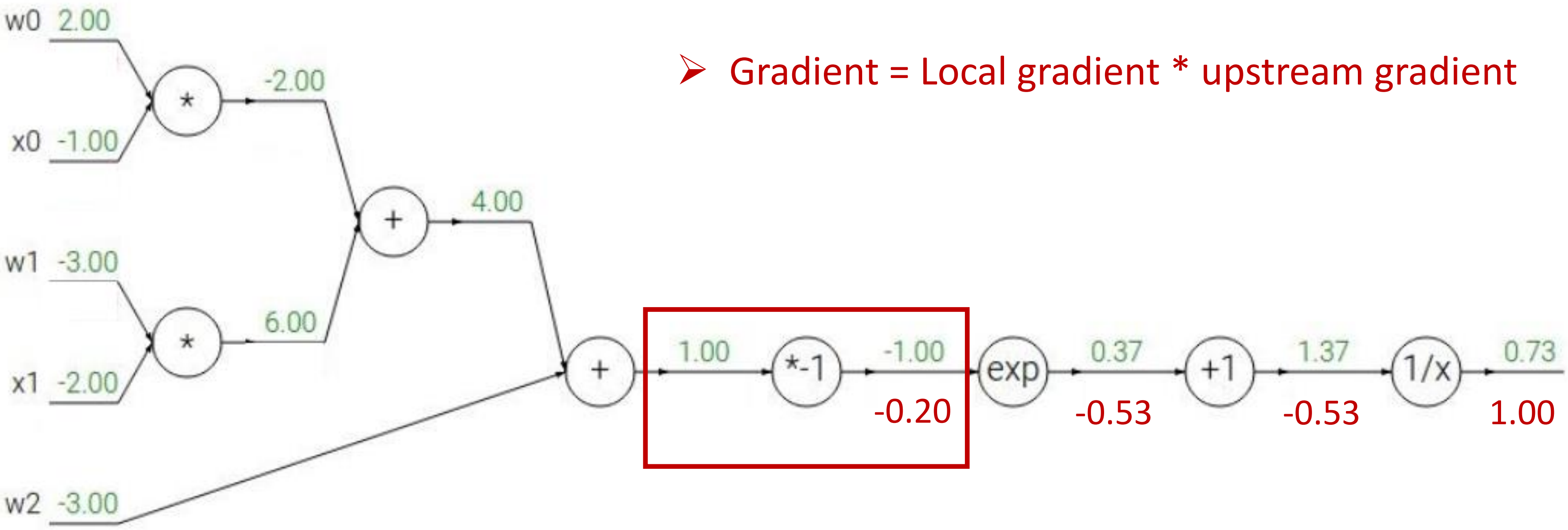
$$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

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

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

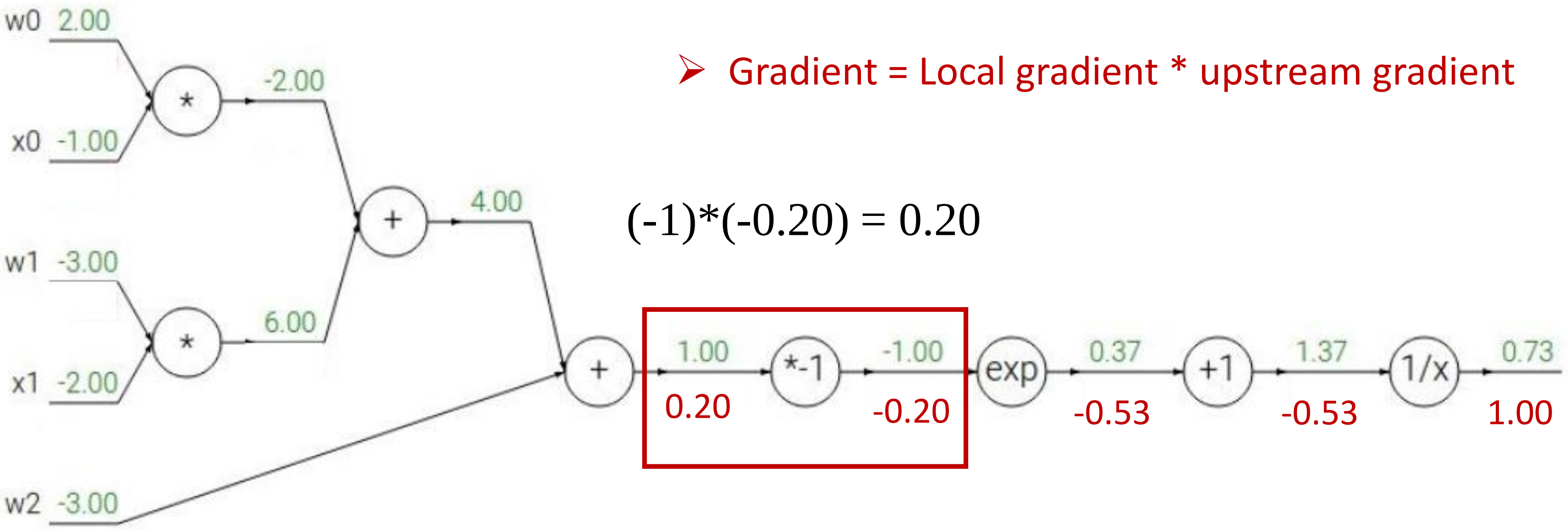
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

(-1)*(-0.20) = 0.20

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

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

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

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

Slide from CS231n

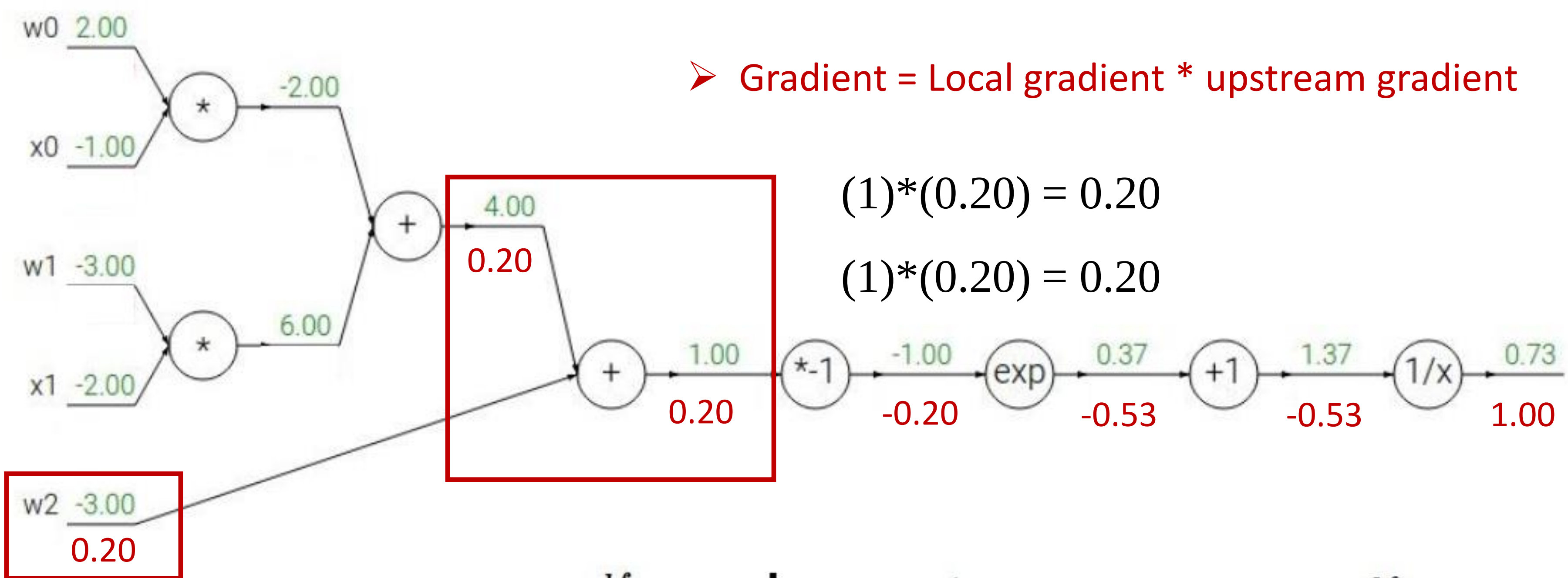
Another Example

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

- Gradient = Local gradient * upstream gradient

(1)*(0.20) = 0.20

(1)*(0.20) = 0.20



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

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

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

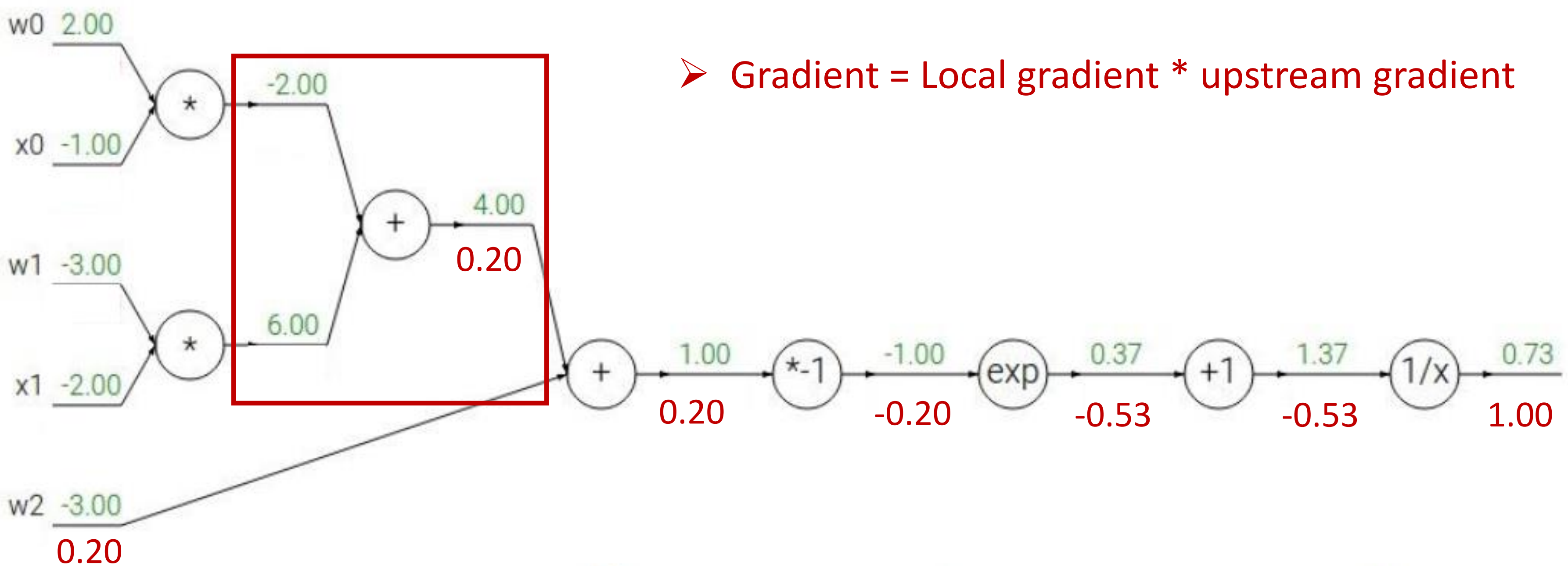
$\boxed{f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1}$

Slide from CS231n

Another Example

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

- Gradient = Local gradient * upstream gradient



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

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

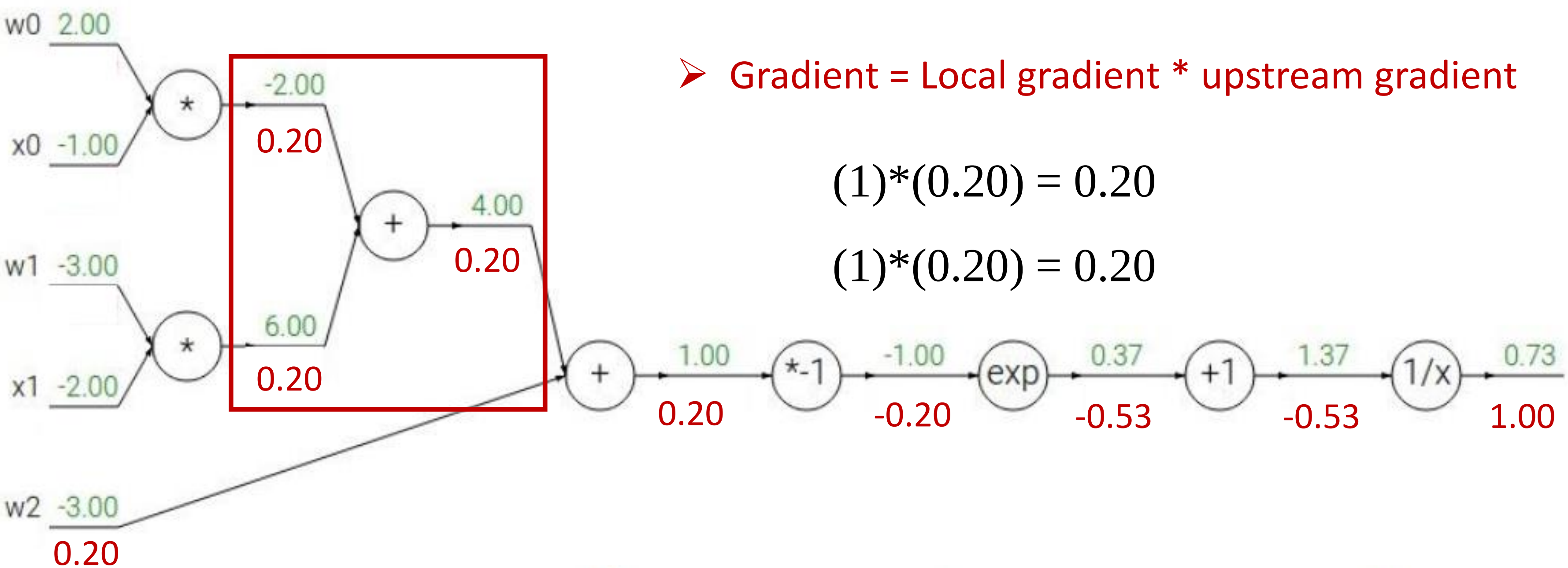
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

(1)*(0.20) = 0.20

(1)*(0.20) = 0.20

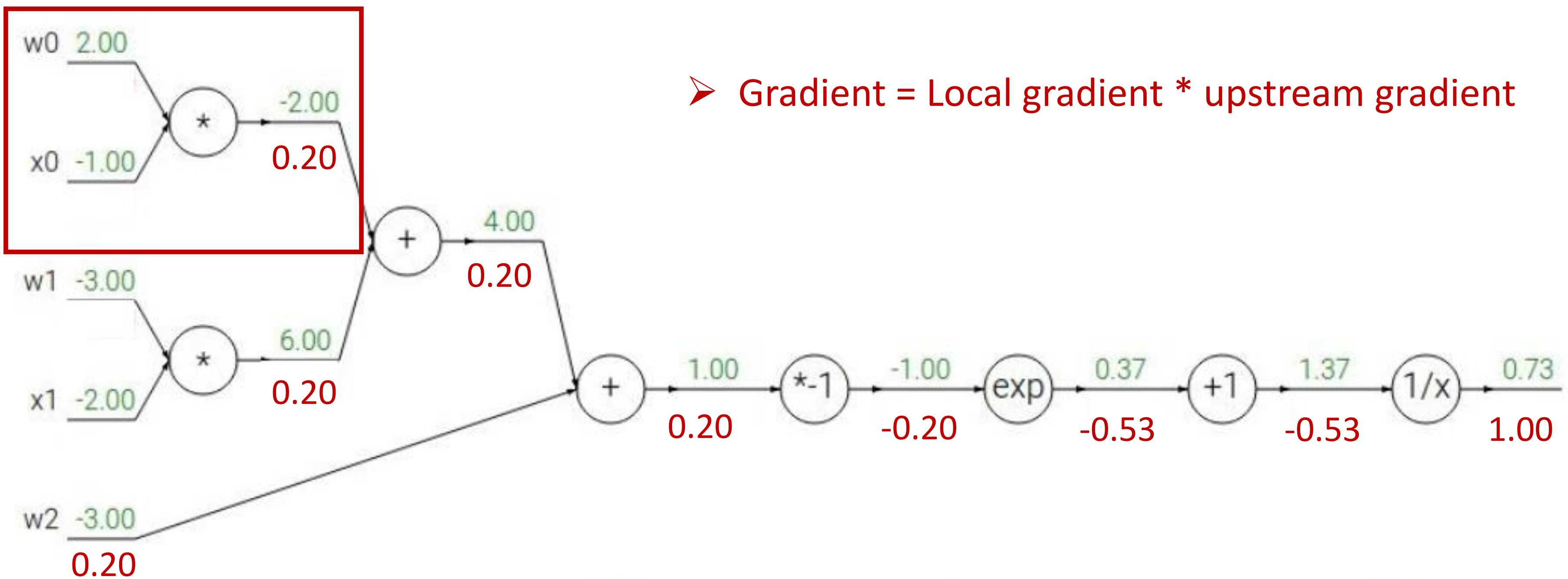
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad \boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1}$$

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

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

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

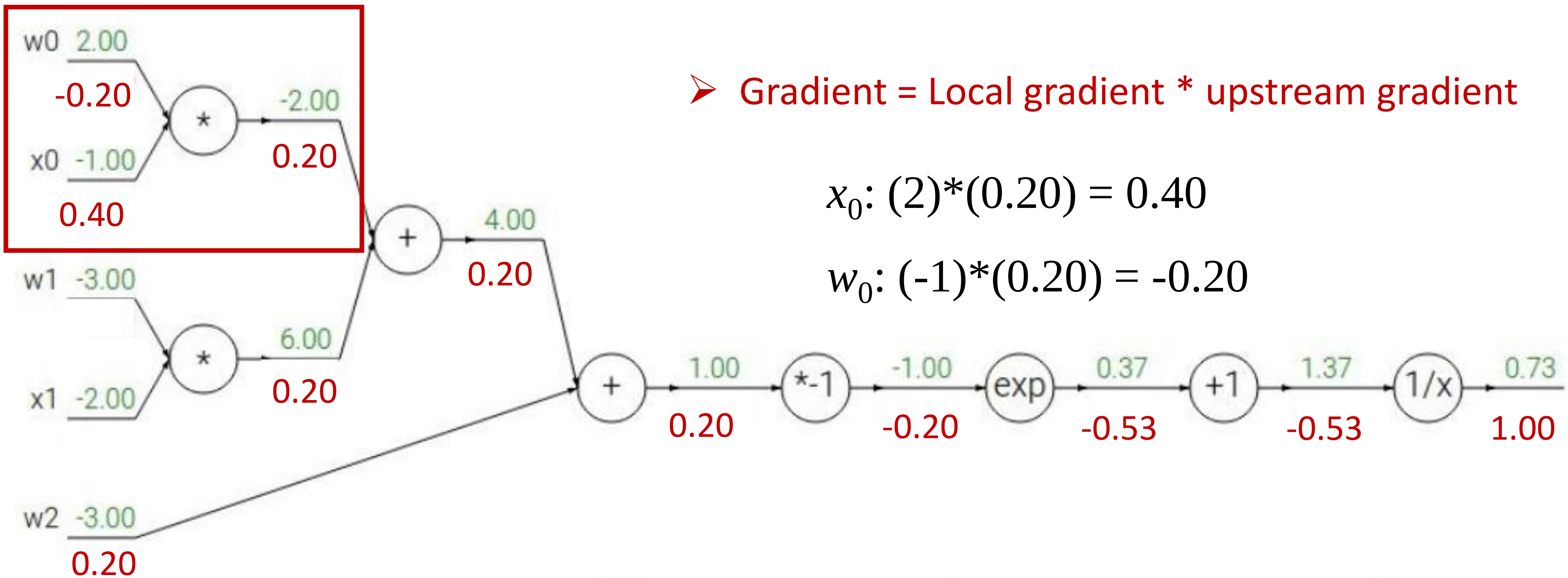
$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

x_0: (2)*(0.20) = 0.40

w_0: (-1)*(0.20) = -0.20

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

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

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

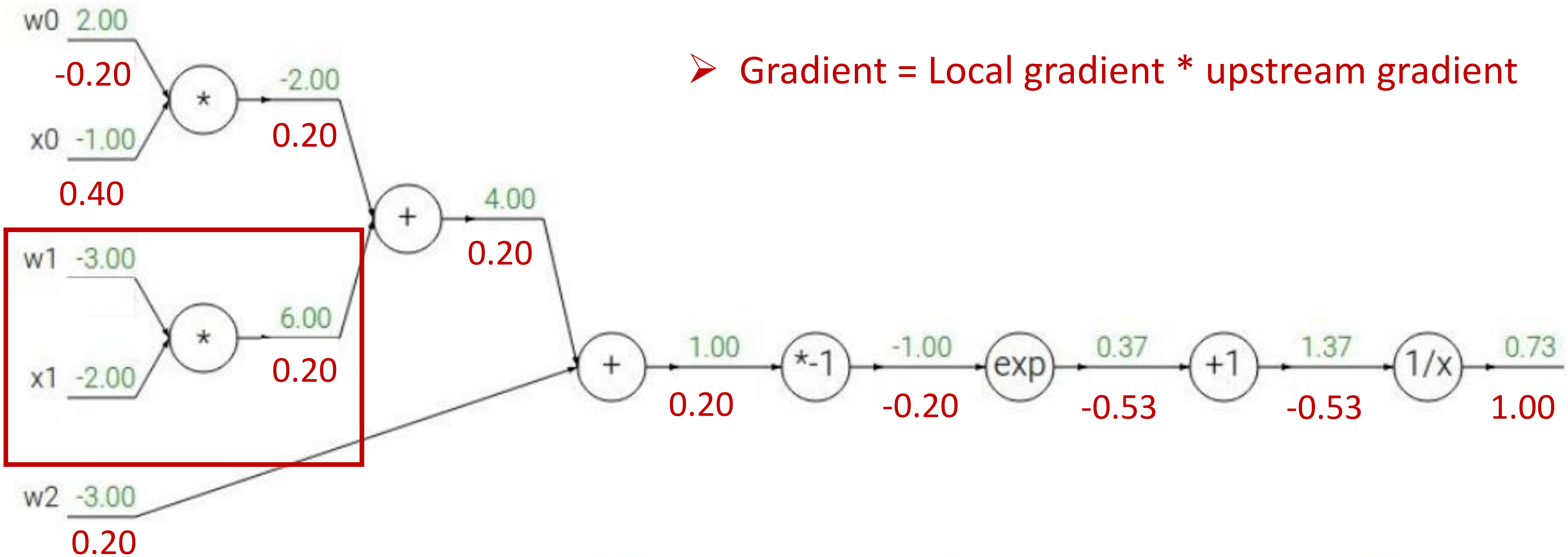
$$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$$

Slide from CS231n

Another Example

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

- Gradient = Local gradient * upstream gradient



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

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

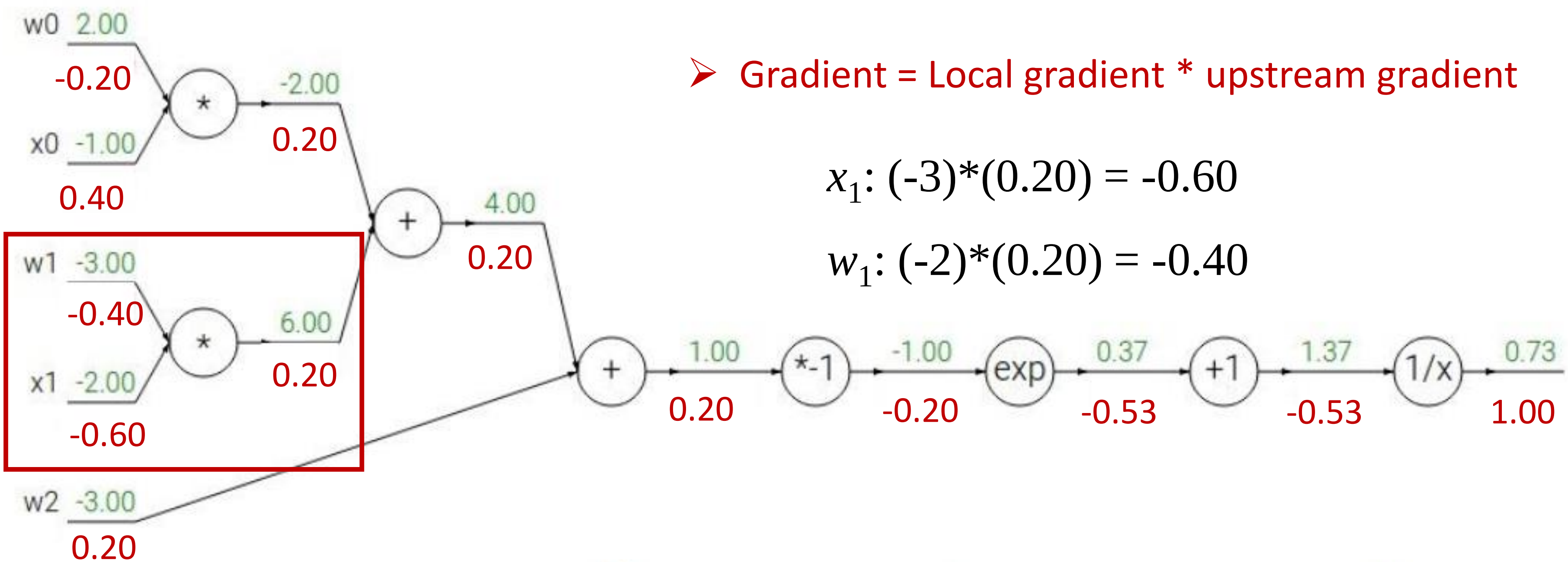
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Slide from CS231n

Another Example

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



- Gradient = Local gradient * upstream gradient

x_1: (-3)*(0.20) = -0.60

w_1: (-2)*(0.20) = -0.40

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

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

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

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

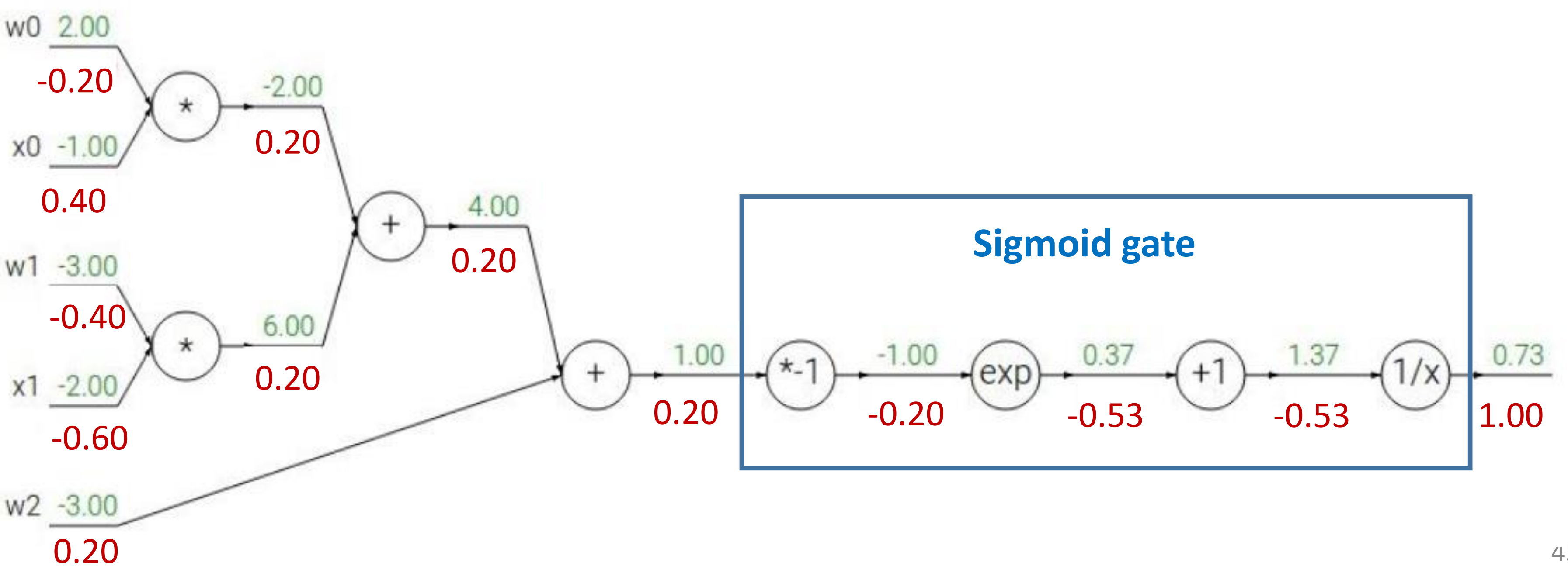
Slide from CS231n

Sigmoid Function

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

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

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



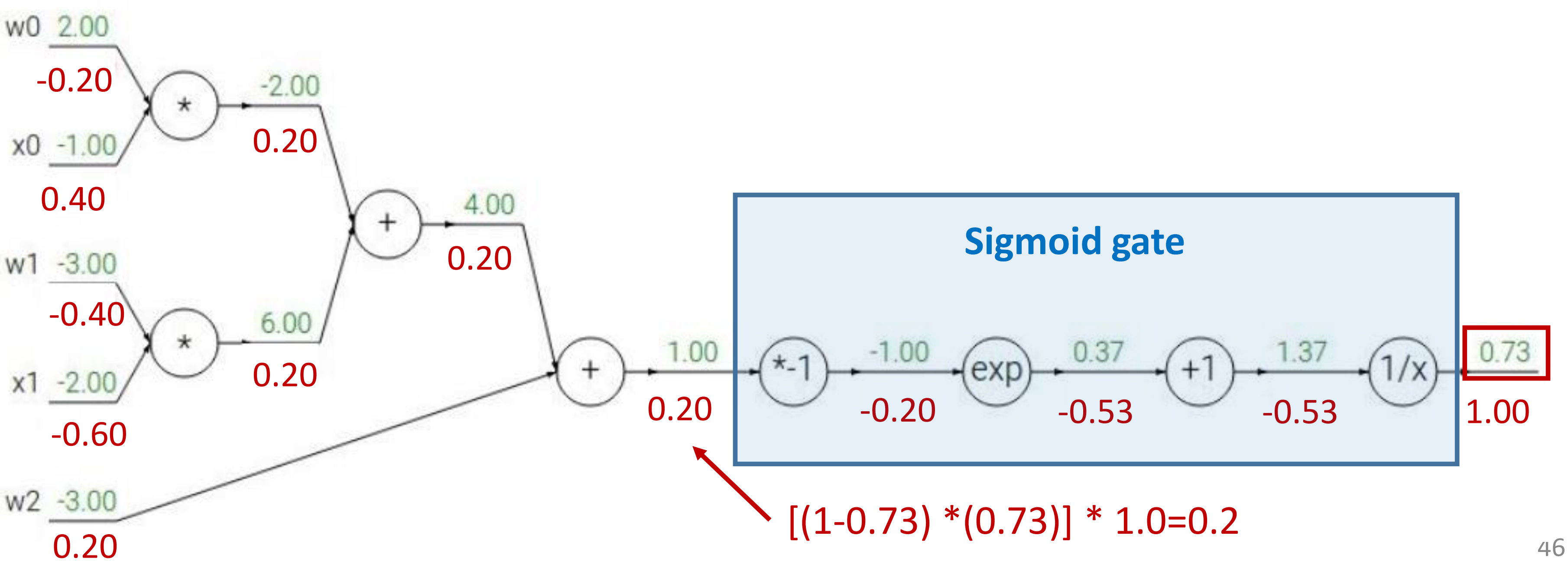
Slide from CS231n

Sigmoid Function

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

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

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



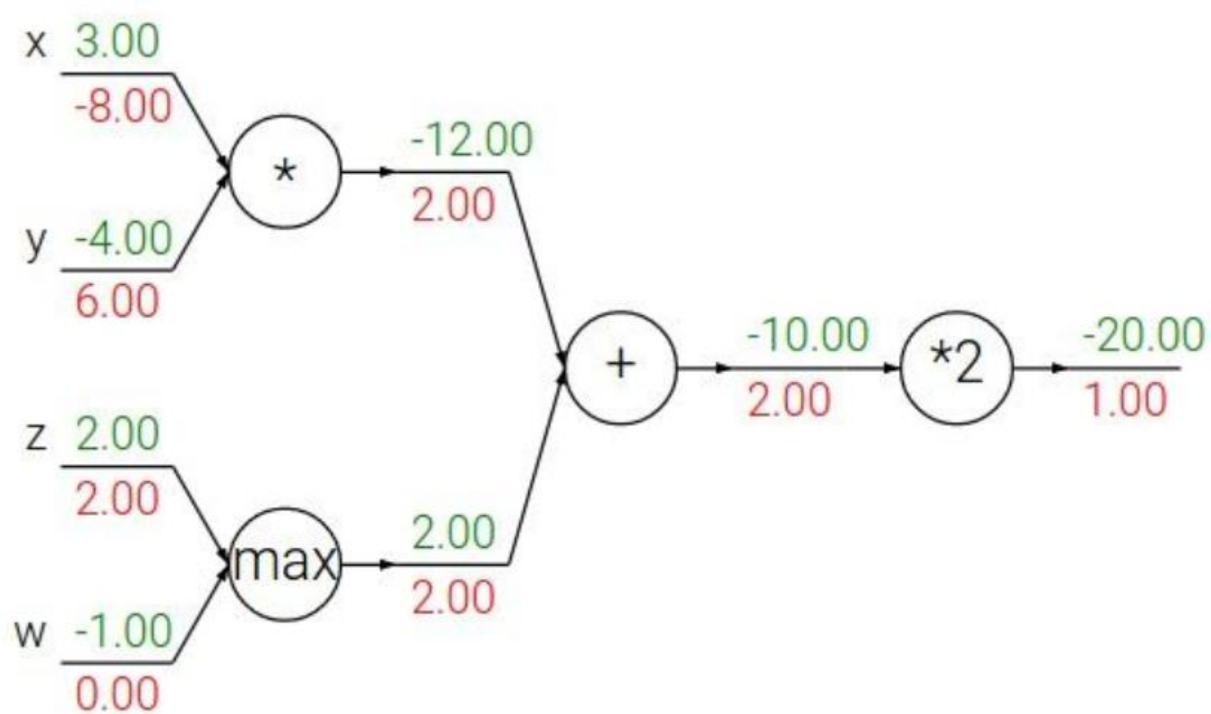
Slide from CS231n

Patterns in Backward Flow

add gate: gradient distributor

max gate: gradient router

mul gate: gradient “switcher”



max(-1, 2) = 2

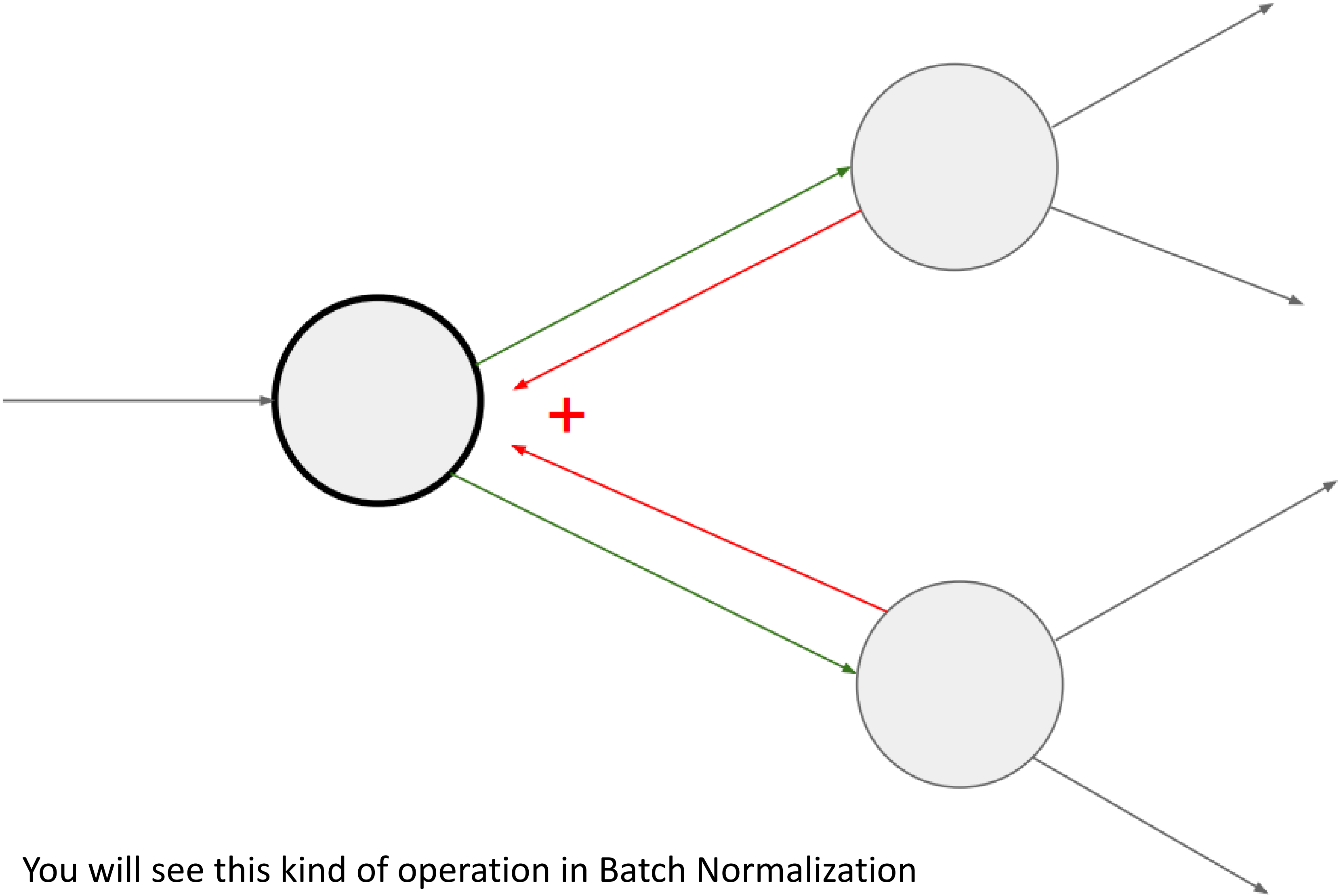
max(-1+δ, 2) = 2

max(-1, 2+δ) = 2+δ

Question #1: what will happen in max gate when two inputs are the same?

Slide from CS231n

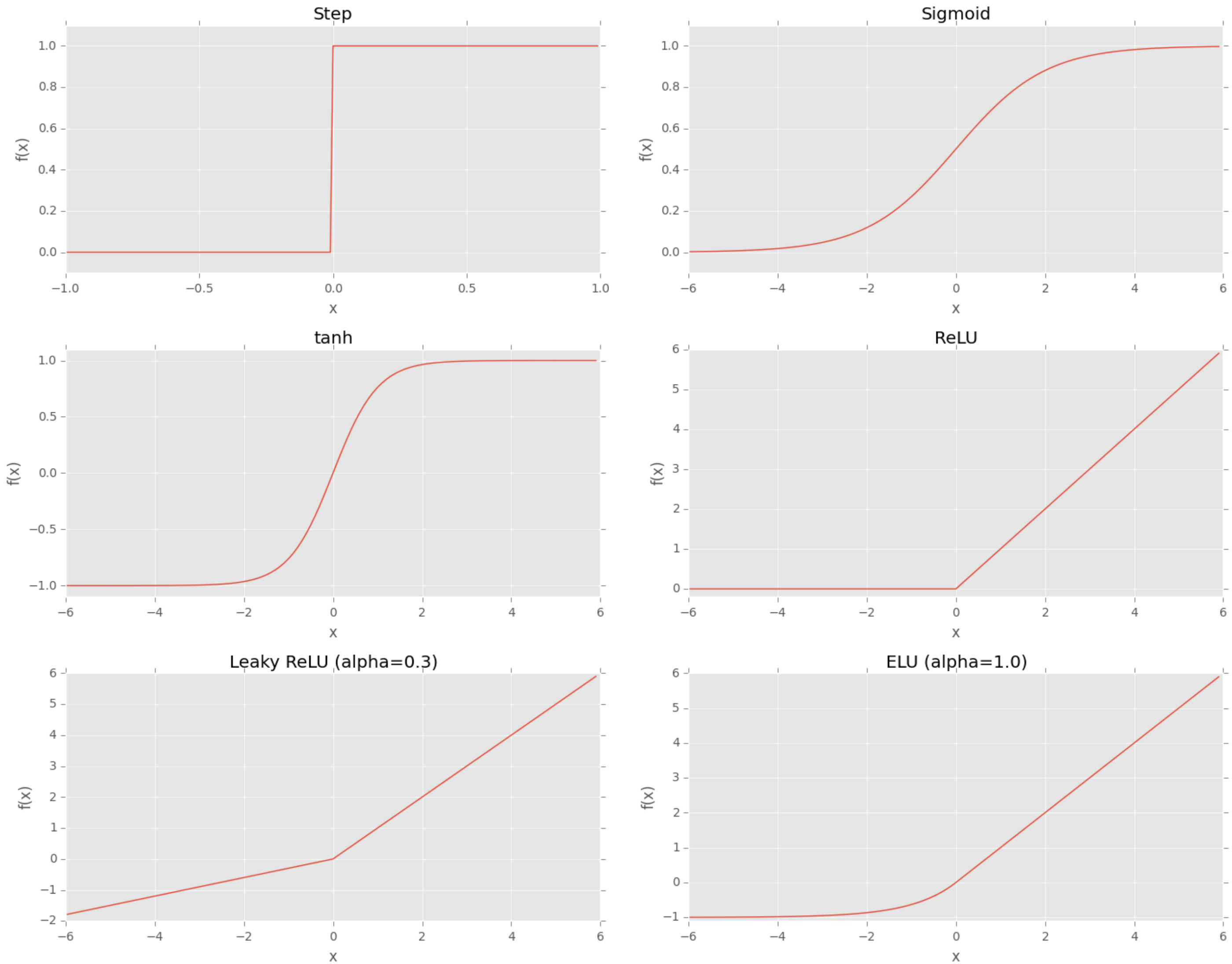
Gradients Add at Branches



- You will see this kind of operation in Batch Normalization

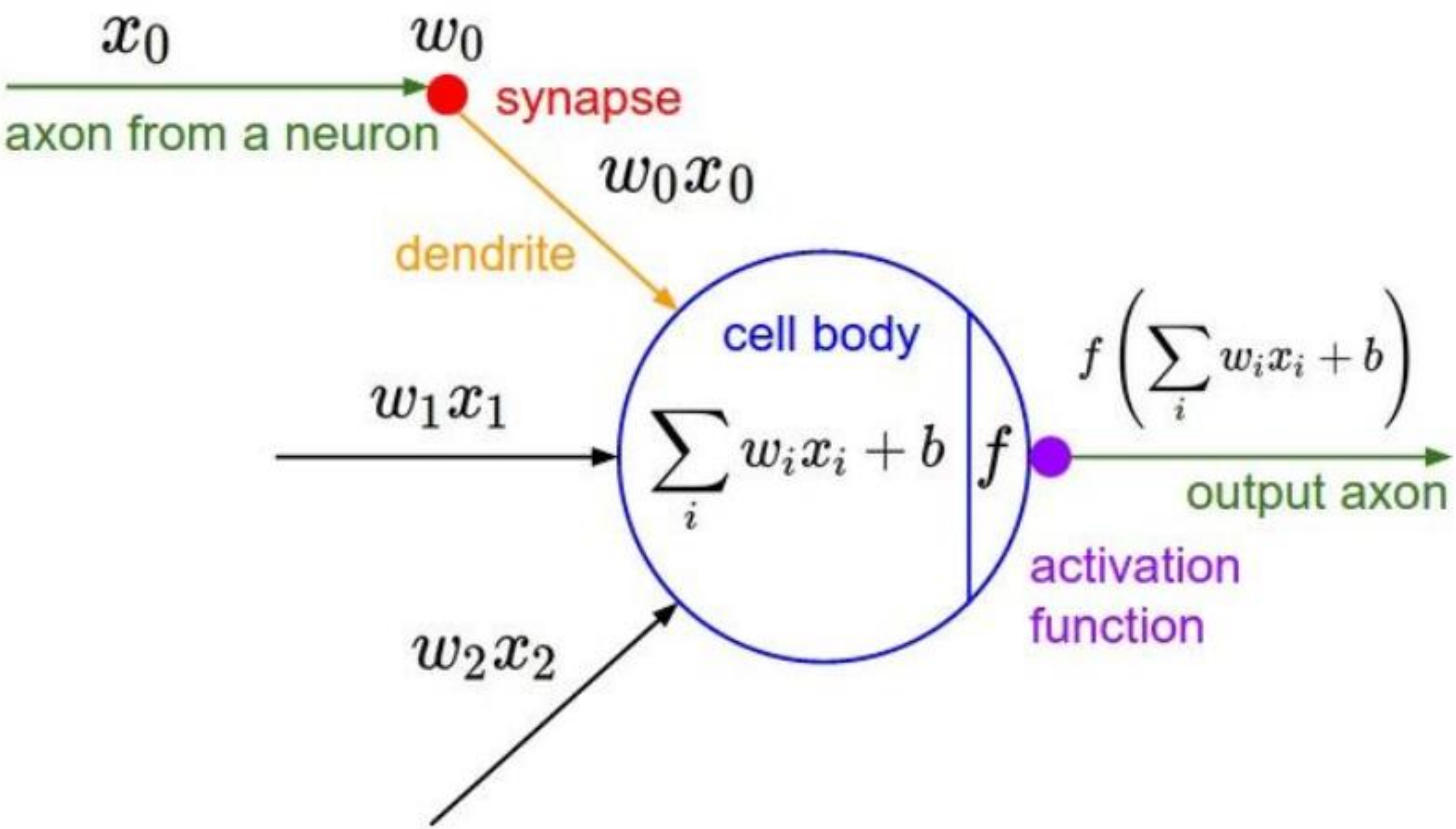
Slide from CS231n

Activation functions



Activation Functions

Activation Functions

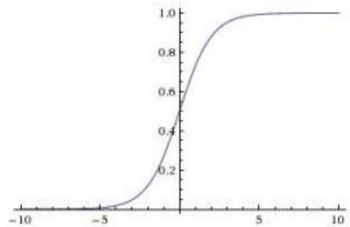


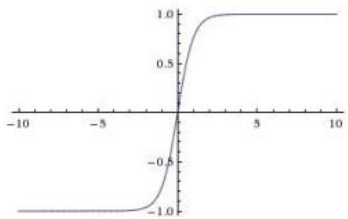
Slide from CS231n

Activation Functions

Sigmoid

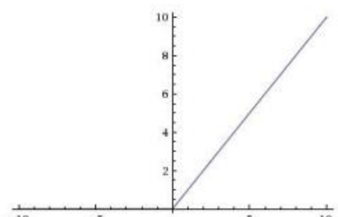
$$\sigma(x) = 1/(1+e^{-x})$$





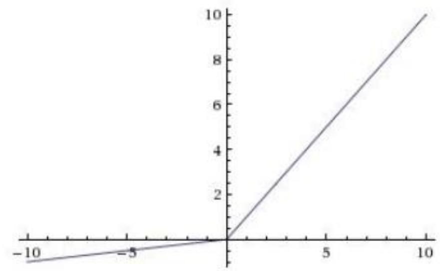
tanh tanh(x)

ReLU max(0,x)



Leaky ReLU

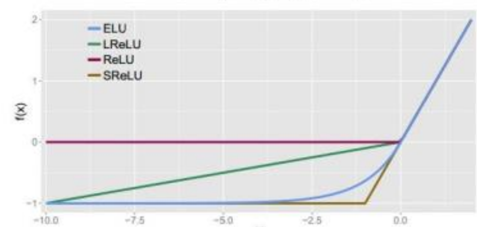
max(0.1x, x)



Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

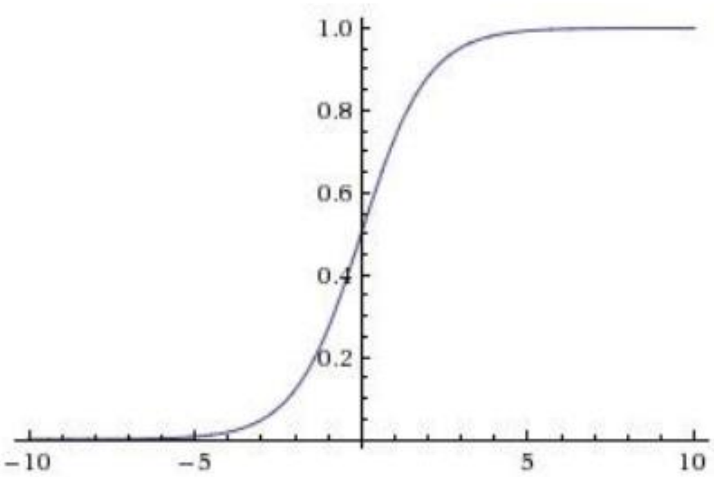
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$



Slide from CS231n

Activation Functions - Sigmoid

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



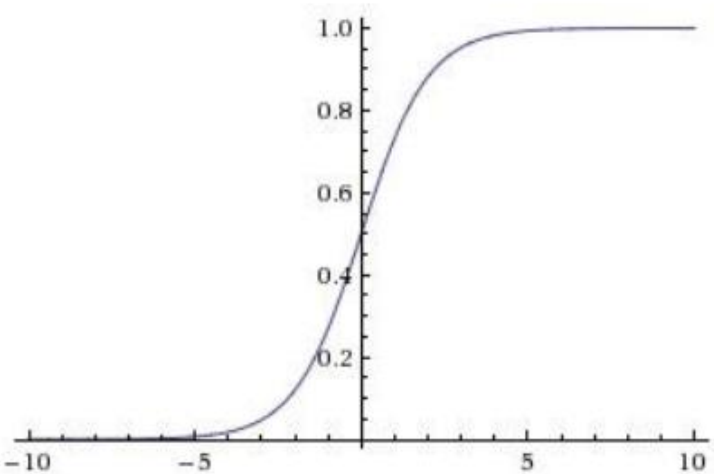
Sigmoid

- Squashes numbers to range [0, 1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

Slide from CS231n

Activation Functions - Sigmoid

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



Sigmoid

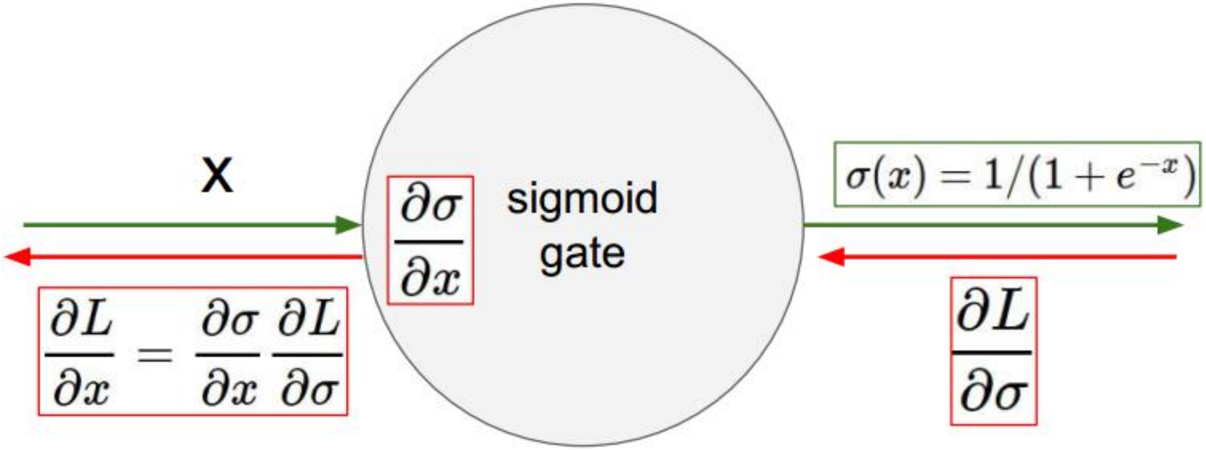
- Squashes numbers to range [0, 1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

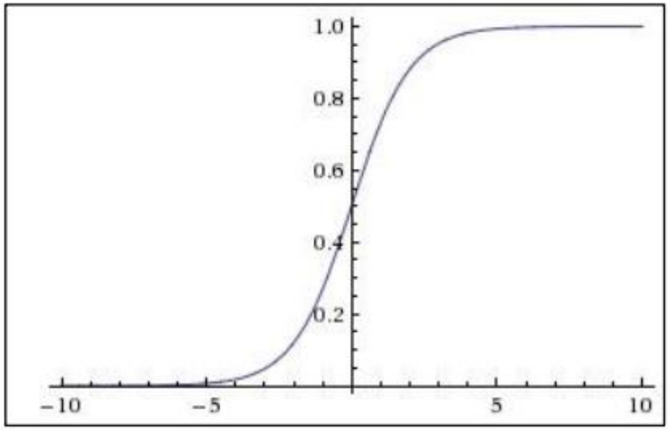
3 problems:

1. Saturated neurons "kill" the gradients

Slide from CS231n

Activation Functions - Sigmoid





What happens when x = -10?

What happens when x = 0?

What happens when x = 10?

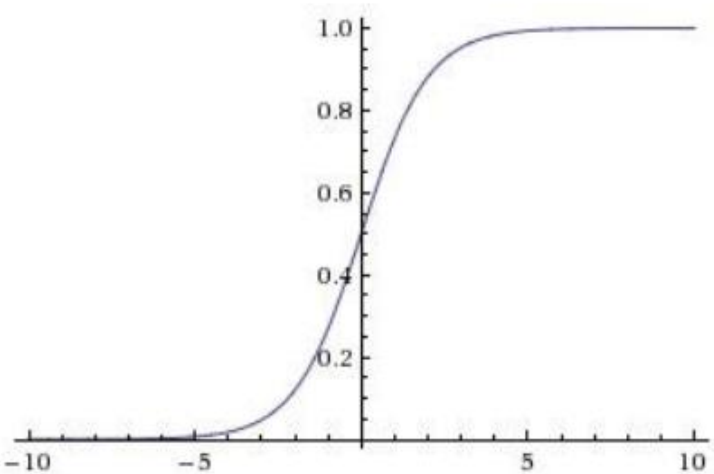
Vanishing gradient problem

SGD:

W += -learning_rate * d**W**

Slide from CS231n

Activation Functions - Sigmoid



Sigmoid

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

- Squashes numbers to range [0, 1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

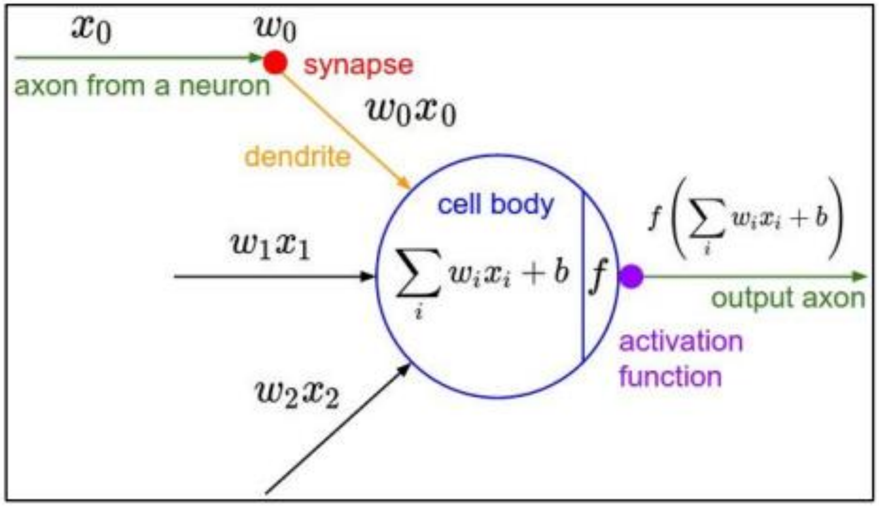
3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered

Slide from CS231n

Activation Functions - Sigmoid

- <u>Consider what happens when the input to a neuron (x) is always positive</u>:



$$f\left(\sum_i w_i x_i + b\right)$$

- **Question #1:** what can we say about the gradients on **w**?

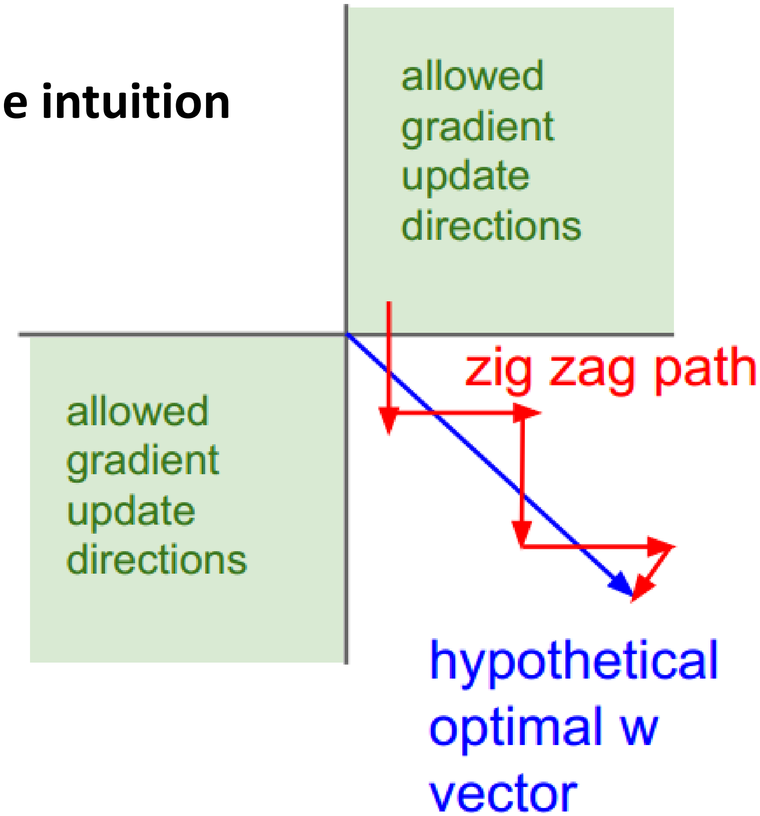
Slide from CS231n

Activation Functions - Sigmoid

- <u>Consider what happens when the input to a neuron (x) is always positive:</u>

$$f\left(\sum_i w_i x_i + b\right)$$

Simple intuition



(this is also why we want to zero-mean data!)

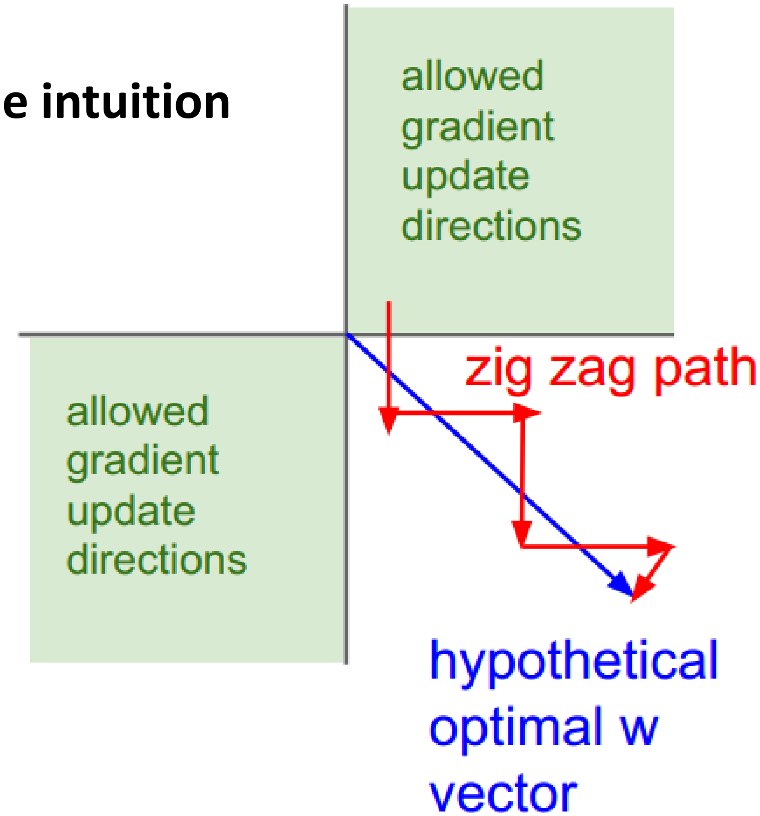
Slide from CS231n

Activation Functions - Sigmoid

- <u>Consider what happens when the input to a neuron (x) is always positive:</u>

$$f\left(\sum_i w_i x_i + b\right)$$

Simple intuition



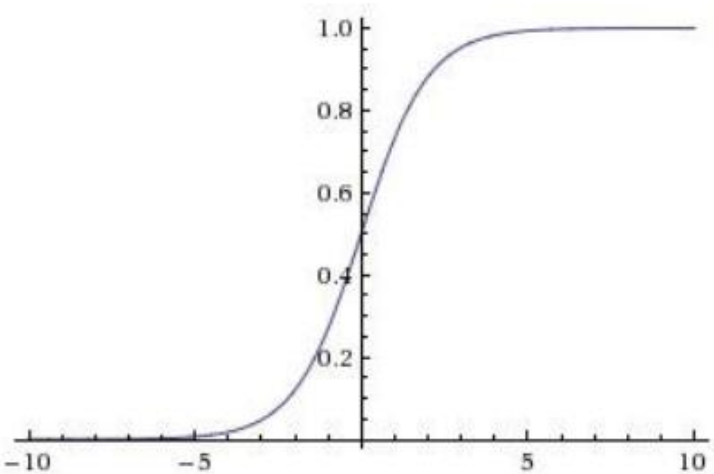
- **Not zero centered data have slower convergence**

(this is also why we want to zero-mean data!)

Slide from CS231n

Activation Functions - Sigmoid

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



Sigmoid

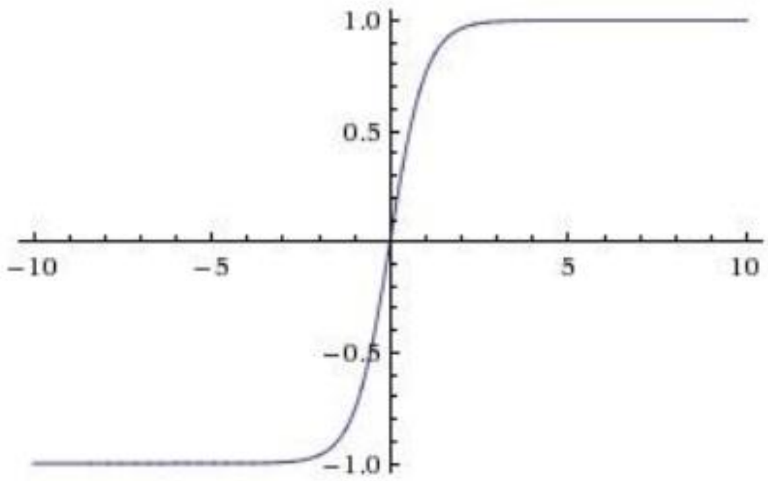
- Squashes numbers to range [0, 1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. Exp() is a bit compute expensive

Slide from CS231n

Activation Functions - Tanh



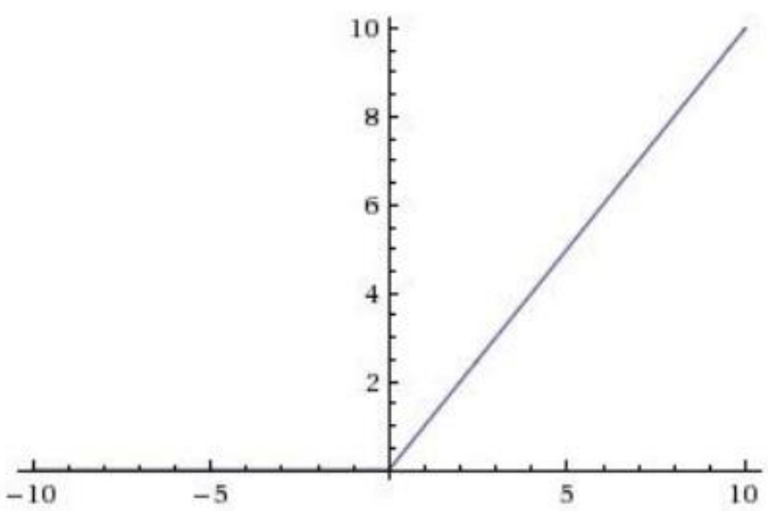
Tanh(x)

- Squashes numbers to range [-1,1]
- Zero centered (nice)
- Still kills gradients when saturated

[LeCun et al., 1991]

Activation Functions - ReLU

- ReLU (Rectified Linear Unit)



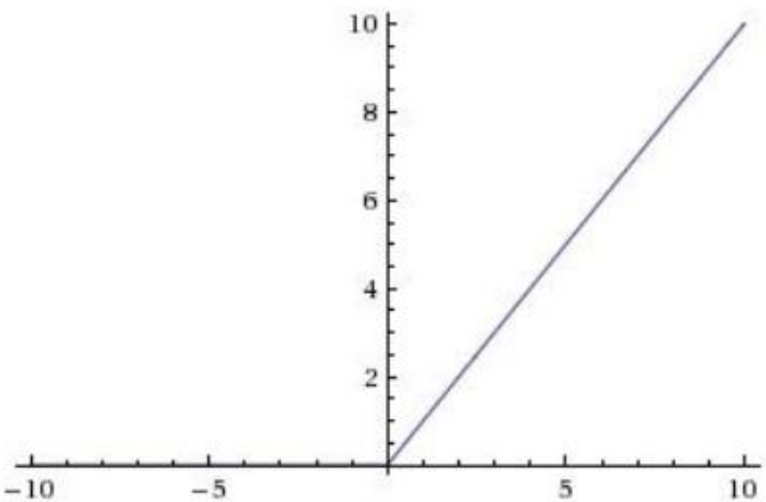
- Computes **f(x) = max(0, x)**
- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

- ReLU is the **default recommendation** what you should use.

[Krizhevsky et al., 2012]

Activation Functions - ReLU

- ReLU (Rectified Linear Unit)



- Computes **f(x) = max(0, x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

Problems:

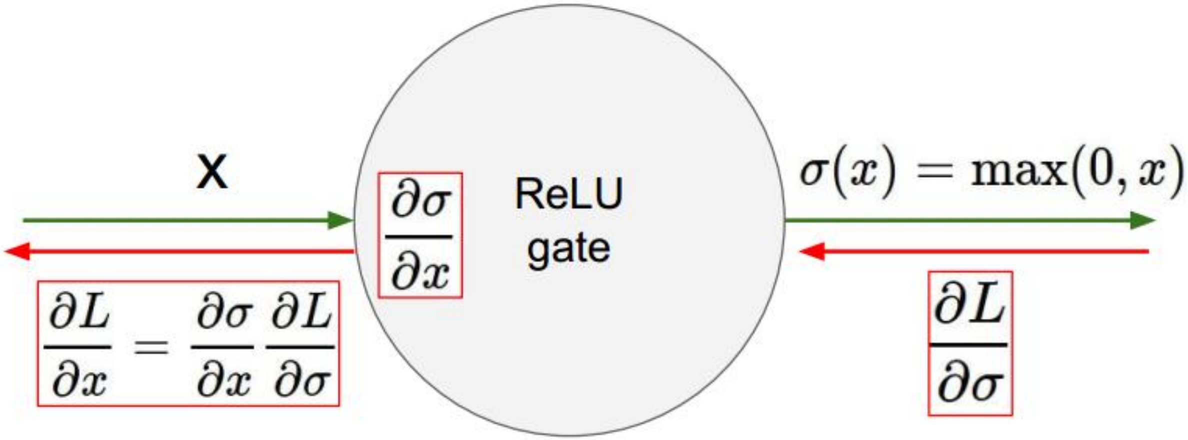
- Not zero-centered output
- An annoyance:

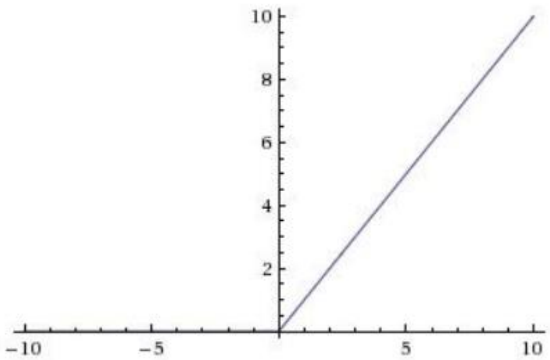
hint: what is the gradient when x < 0?

- ReLU is the **default recommendation** what you should use.

[Krizhevsky et al., 2012]

Activation Functions - ReLU





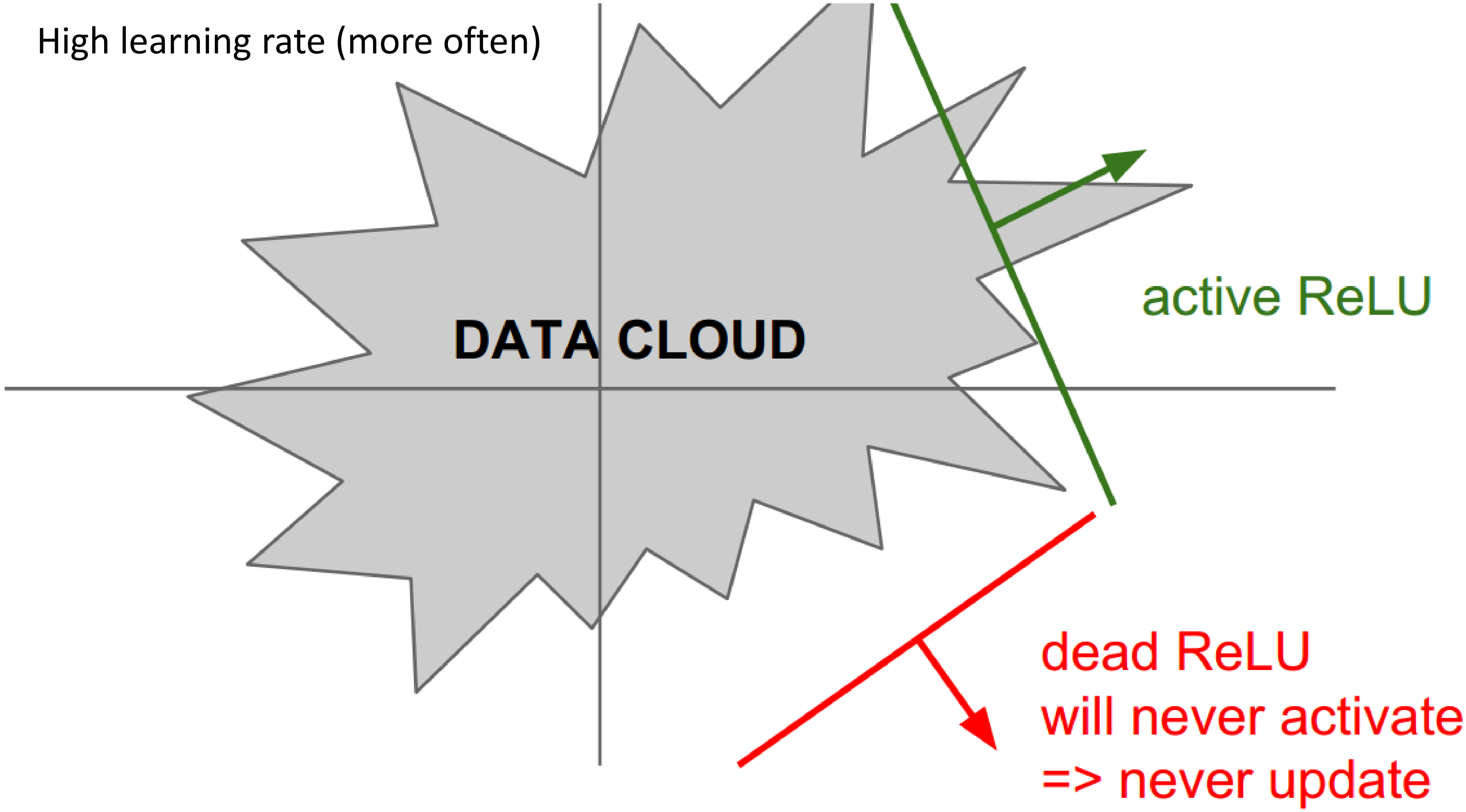
- What happens when x = -10? => **Dead ReLU**
- What happens when x = 0?
- What happens when x = 10?

Slide from CS231n

Activation Functions - ReLU

Dead ReLU:

- Initialization (unlucky)
- High learning rate (more often)

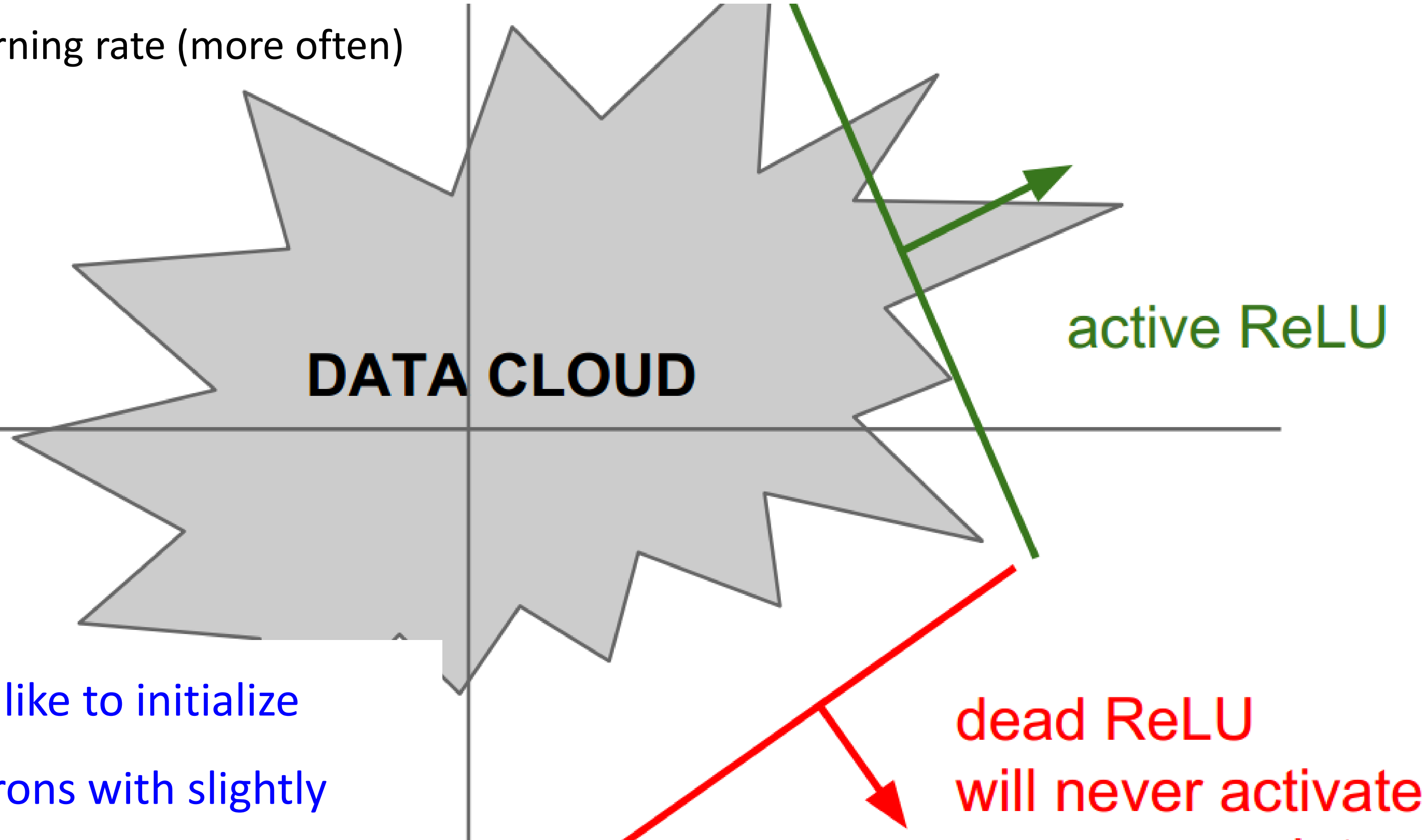


Slide from CS231n

Activation Functions - ReLU

Dead ReLU:

- Initialization (unlucky)
- High learning rate (more often)

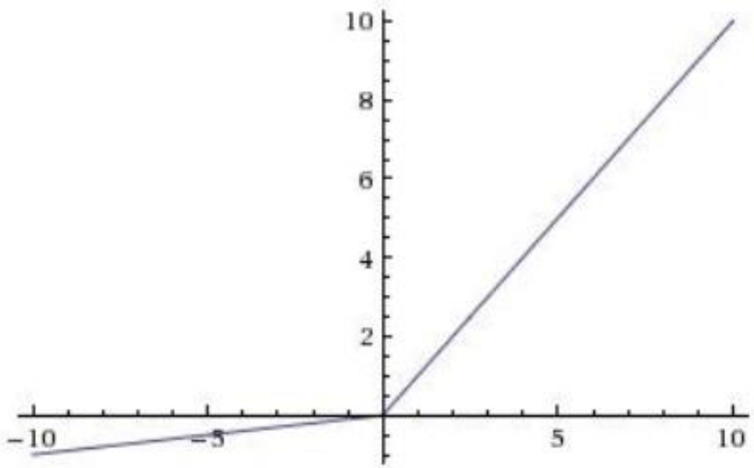


$\Rightarrow$ People like to initialize ReLU neurons with slightly positive biases (e.g. 0.01)

$f(\mathbf{x} + \mathbf{b})$???

Slide from CS231n

Activation Functions – Leaky ReLU



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **Will not "die"**

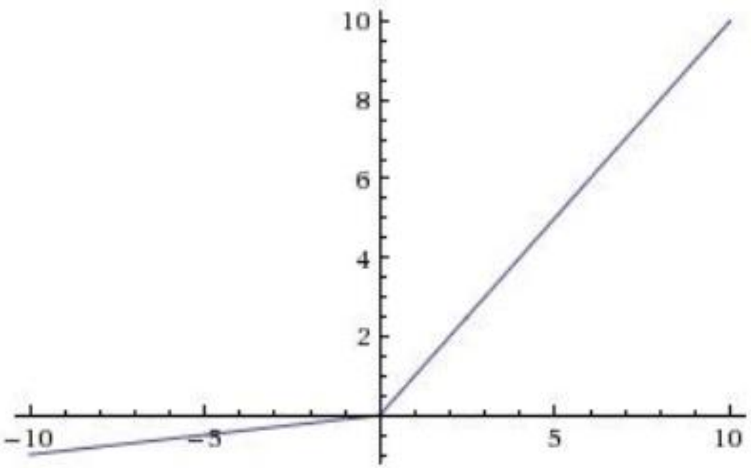
- **Leaky ReLU**

$$f(x) = \max(0.01x, x)$$

[Mass et al., 2013]
[He et al., 2015]

Activation Functions – Leaky ReLU

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **Will not "die"**



- **Leaky ReLU**

$$f(x) = \max(0.01x, x)$$

- **Parametric Rectifier (PReLU)**

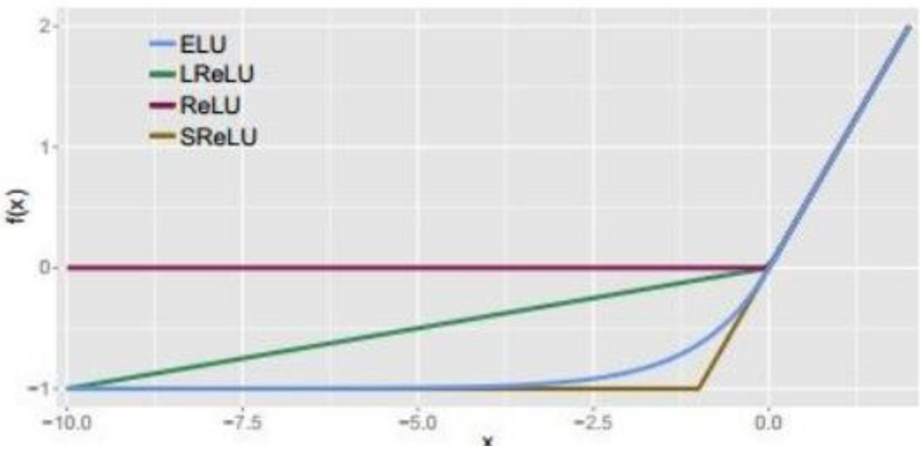
$$f(x) = \max(\alpha x, x)$$

Backprop into α to learn!

[Mass et al., 2013]
[He et al., 2015]

Activation Functions – ELU

- **Exponential Linear Units (ELU)**



- All benefits of ReLU
- Does not die
- Closer to zero mean outputs

- Computation requires exp()

$$f(x)=\begin{cases} x & \text{if } x>0 \\ \alpha(\exp(x)-1) & \text{if } x\le 0 \end{cases}$$

Slide from CS231n

Activation Functions – Maxout "Neuron"

- Very different form of the neuron, it's not just an activation function looks diff erent
- It changes with the neuro compute and how it computes

- Generalizes **ReLU** and **Leaky ReLU**
- Linear Regime! Does not saturate! Does not die!

$$\max\left(w_1^T x + b_1, w_2^T x + b_2\right)$$

- **Problem:** doubles the number of parameters/neuron ☹

[Goodfellow et al., 2013]

Summary

In practice:

- Use **ReLU**. Be careful with your learning rates
- Try out **Leaky ReLU / Maxout / ELU**
- Try out **tanh** but don't expect much
- Don't use **sigmoid,** tanh is better than it

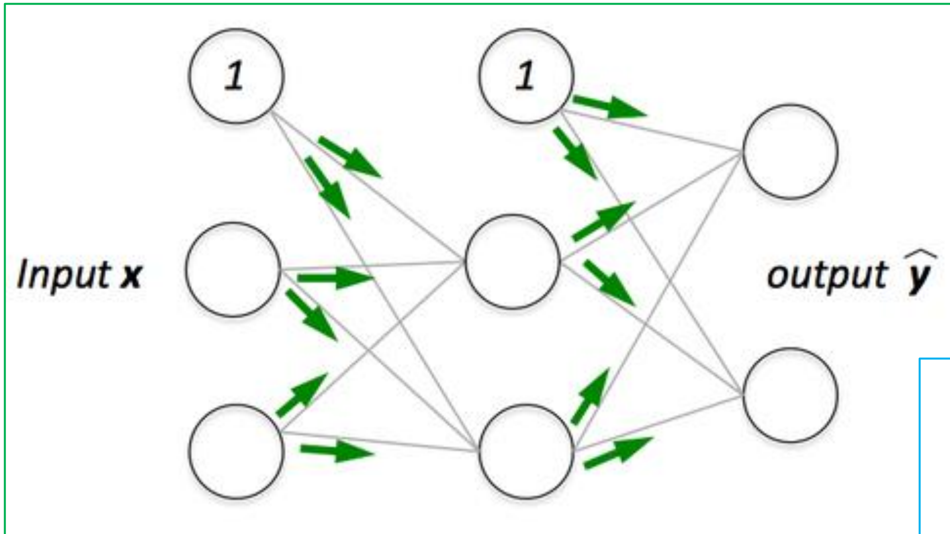
- RNN / LSTM still uses sigmoid, but there are specific reason…

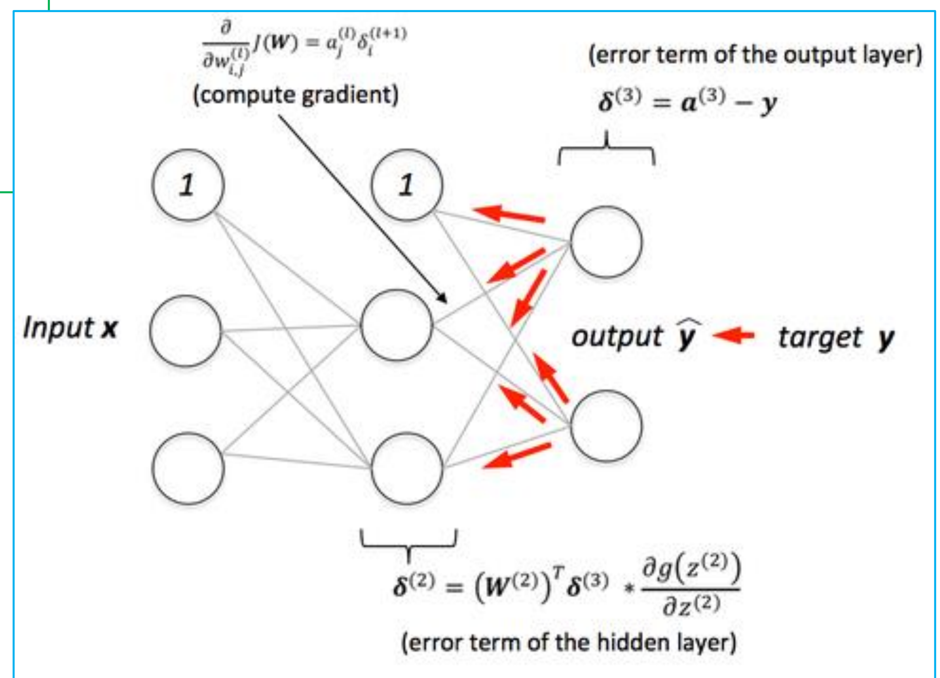
Slide from CS231n

참고자료

Backpropagation

- Apply the back propagation to update the NN





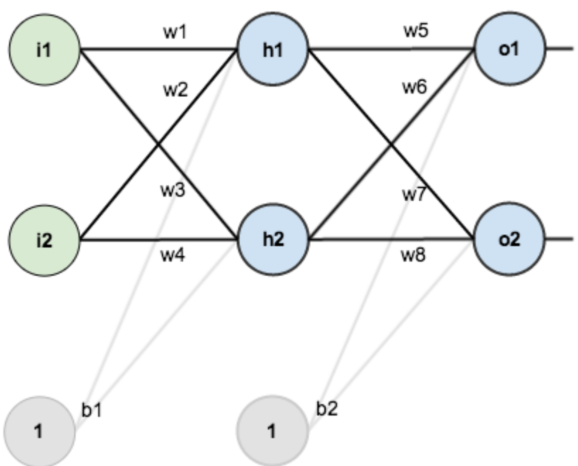
Chain rule:

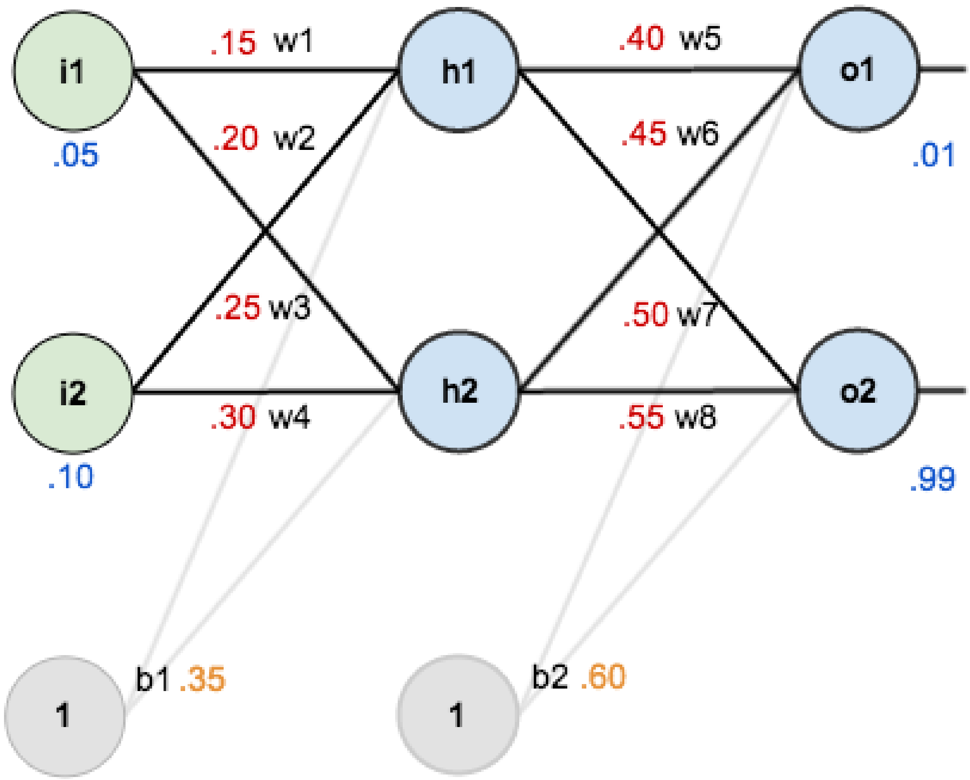
$$f(g(x)) = y$$

$$\frac{\partial y}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

Basic Structure

In order to have some numbers to work with, here's are the initial weights, the biases, and training inputs/outputs:





The Forward Pass

We figure out the *total net input* to each hidden layer neuron, *squash* the total net input using an *activation function* (here we use the *logistic function*), then repeat the process with the output layer neurons.

Here's how we calculate the total net input for h_1:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

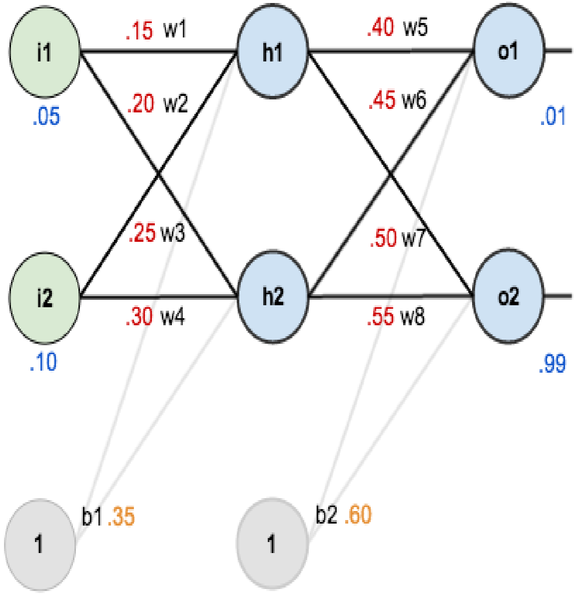
$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

We then squash it using the logistic function to get the output of h_1:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$



The Forward Pass

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Here's the output for o_1:

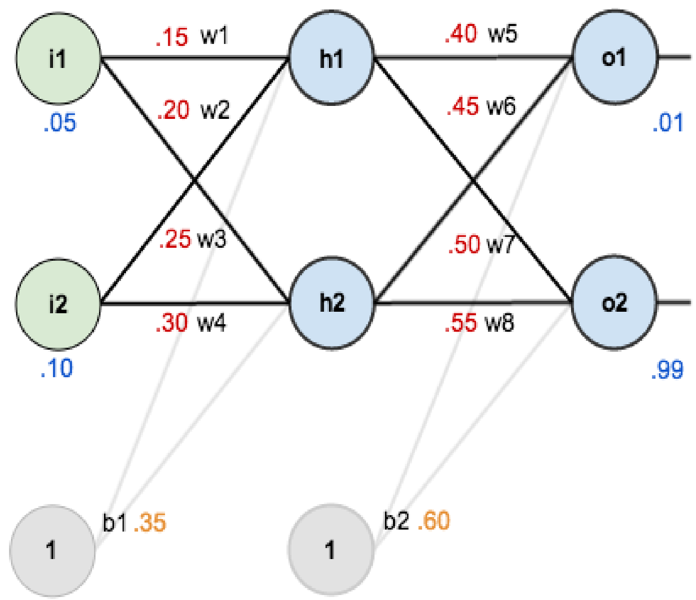
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$



The Forward Pass

Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Some sources refer to the target as the *ideal* and the output as the *actual*.

The $\frac{1}{2}$ is included so that exponent is cancelled when we differentiate later on. The result is eventually multiplied by a learning rate anyway so it doesn't matter that we introduce a constant here [1].

The Forward Pass

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

The Backwards Pass

Output Layer

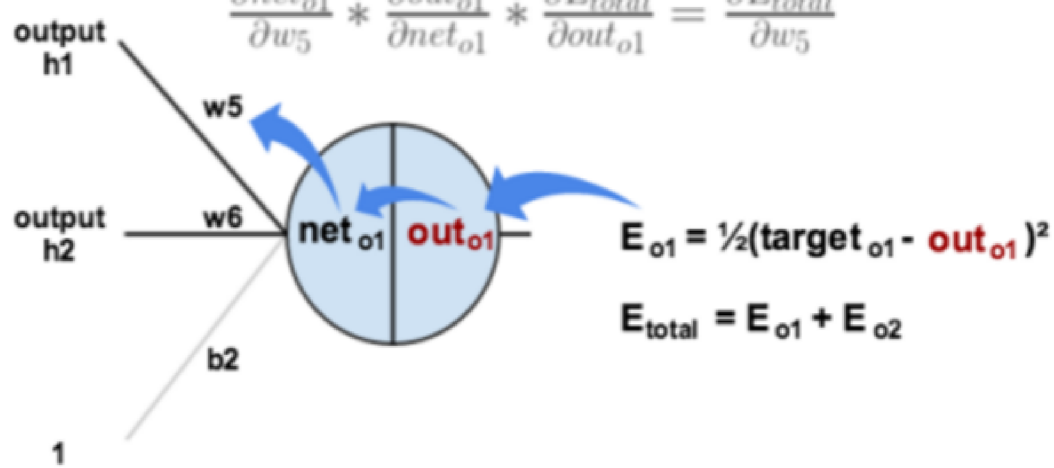
Consider w_5. We want to know how much a change in w_5 affects the total error, aka $\frac{\partial E_{total}}{\partial w_5}$.

> $\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



The Backwards Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

First, how much does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$-(target - out)$ is sometimes expressed as $out - target$

When we take the partial derivative of the total error with respect to out_{o1}, the quantity $\frac{1}{2}(target_{o2} - out_{o2})^2$ becomes zero because out_{o1} does not affect it which means we're taking the derivative of a constant which is zero.

The Backwards Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

The partial derivative of the logistic function is the output multiplied by 1 minus the output:

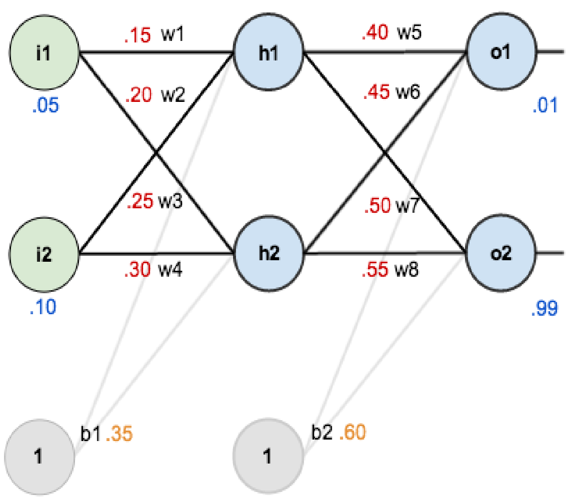
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$



Putting it all together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

The Backwards Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

You'll often see this calculation combined in the form of the delta rule:

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

Alternatively, we have $\frac{\partial E_{total}}{\partial out_{o1}}$ and $\frac{\partial out_{o1}}{\partial net_{o1}}$ which can be written as $\frac{\partial E_{total}}{\partial net_{o1}}$, aka δ_{o1} (the Greek letter delta) aka the *node delta*. We can use this to rewrite the calculation above:

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

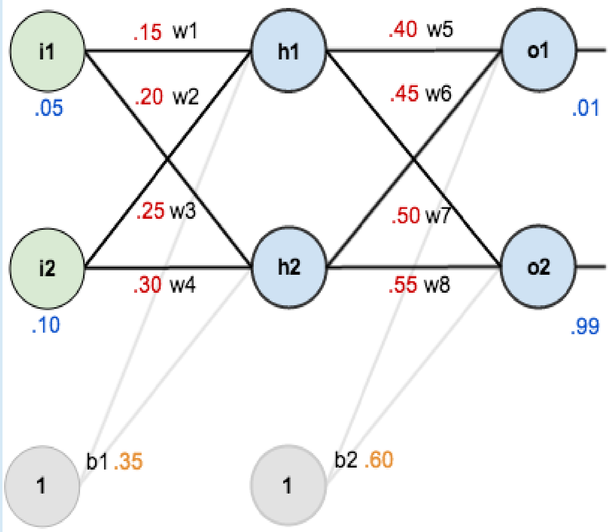
$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

Therefore:

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

Some sources extract the negative sign from δ so it would be written as:

$$\frac{\partial E_{total}}{\partial w_5} = -\delta_{o1} out_{h1}$$



The Backwards Pass

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

> Some sources use α (alpha) to represent the learning rate, others use η (eta), and others even use ϵ (epsilon).

We can repeat this process to get the new weights w_6, w_7, and w_8:

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

The Backwards Pass

Hidden Layer

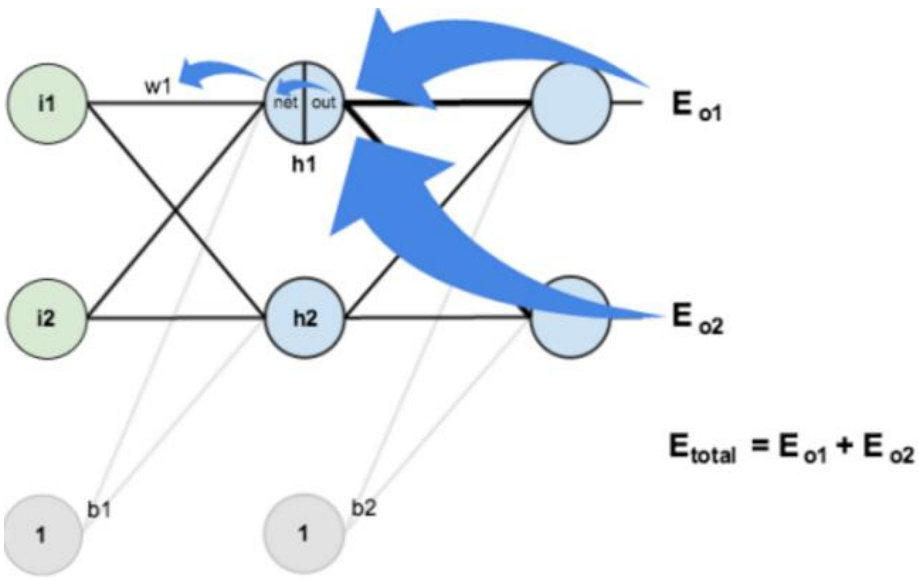
Next, we'll continue the backwards pass by calculating new values for w_1, w_2, w_3, and w_4.

Big picture, here's what we need to figure out:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



The Backwards Pass

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Starting with $\frac{\partial E_{o1}}{\partial out_{h1}}$:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

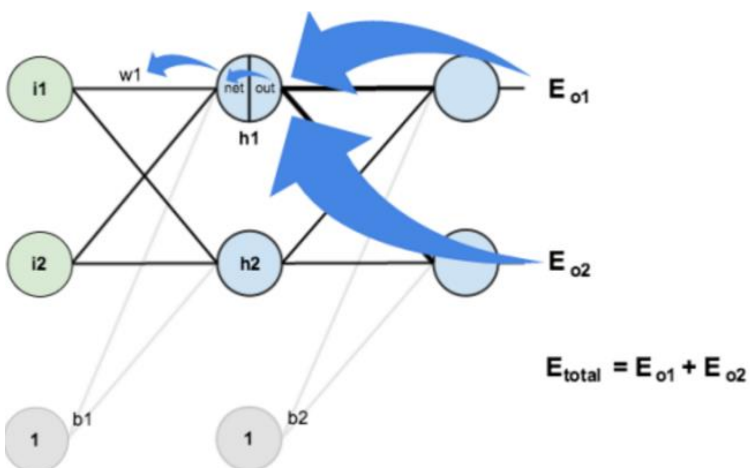
We can calculate $\frac{\partial E_{o1}}{\partial net_{o1}}$ using values we calculated earlier:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

And $\frac{\partial net_{o1}}{\partial out_{h1}}$ is equal to w_5:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$



Plugging them in:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

The Backwards Pass

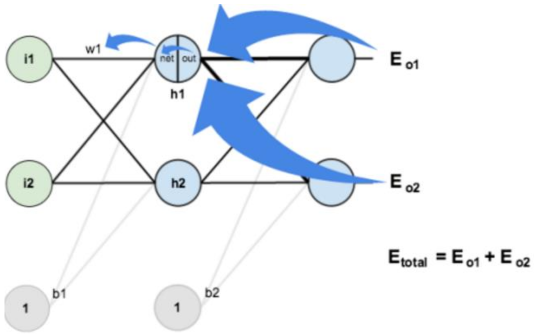
$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

Following the same process for $\frac{\partial E_{o2}}{\partial out_{o1}}$, we get:

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

Therefore:

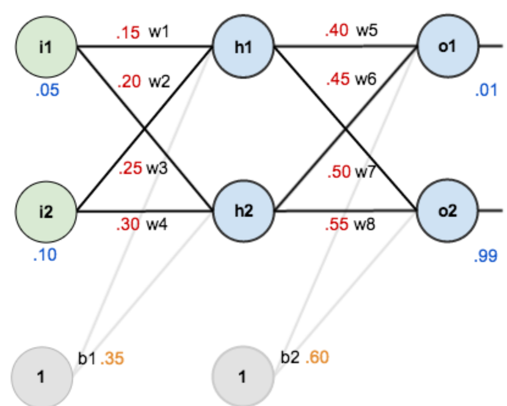
$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$



Now that we have $\frac{\partial E_{total}}{\partial out_{h1}}$, we need to figure out $\frac{\partial out_{h1}}{\partial net_{h1}}$ and then $\frac{\partial net_{h1}}{\partial w}$ for each weight:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$



We calculate the partial derivative of the total net input to h_1 with respect to w_1 the same as we did for the output neuron:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

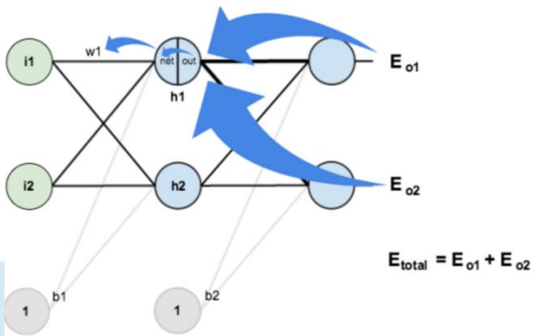
$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

The Backwards Pass

Putting it all together:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

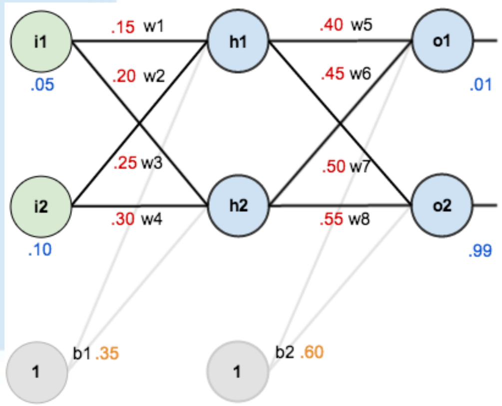


You might also see this written as:

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \delta_o * w_{ho}\right) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$



We can now update w_1:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

The Backwards Pass

Repeating this for w_2, w_3, and w_4

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$