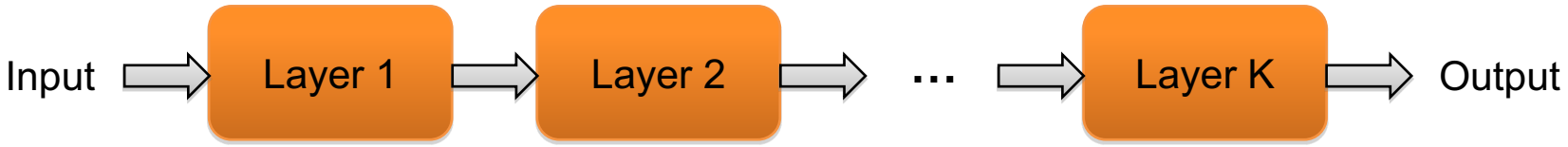
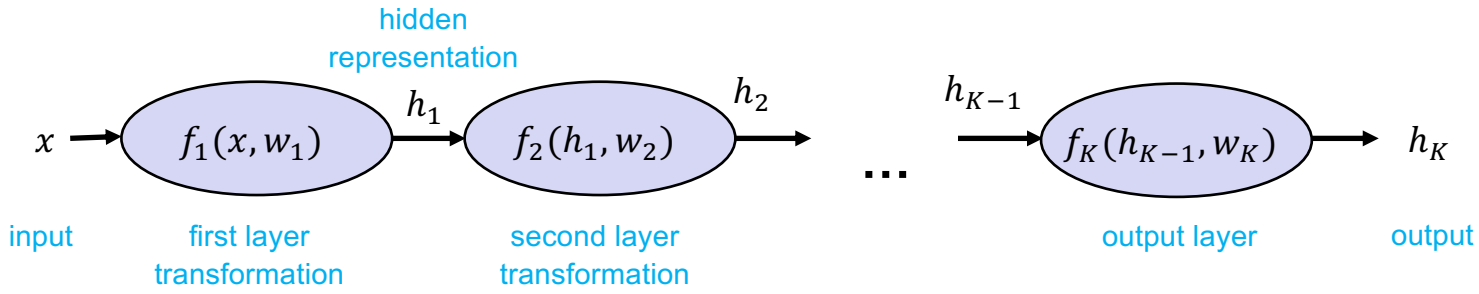
How to train a multi-layer network?

Recall: Multi-layer neural networks

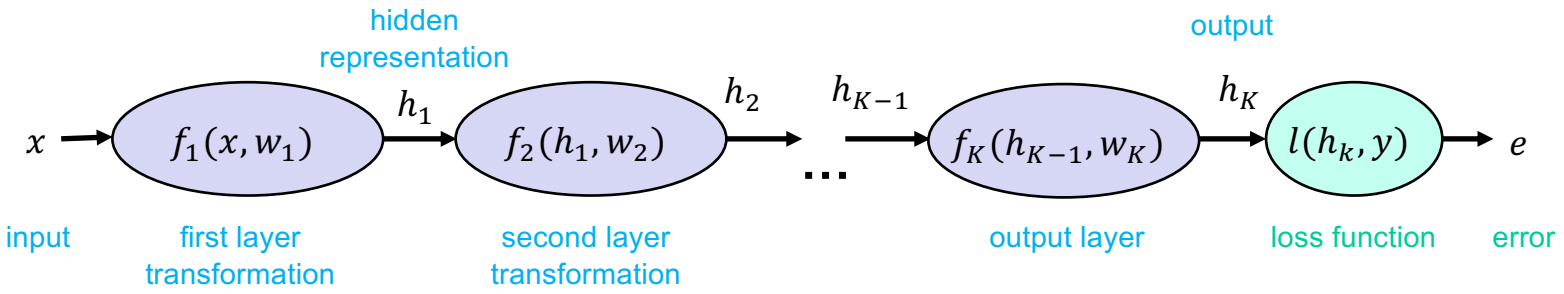
- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):



- More precisely:



Training a multi-layer network

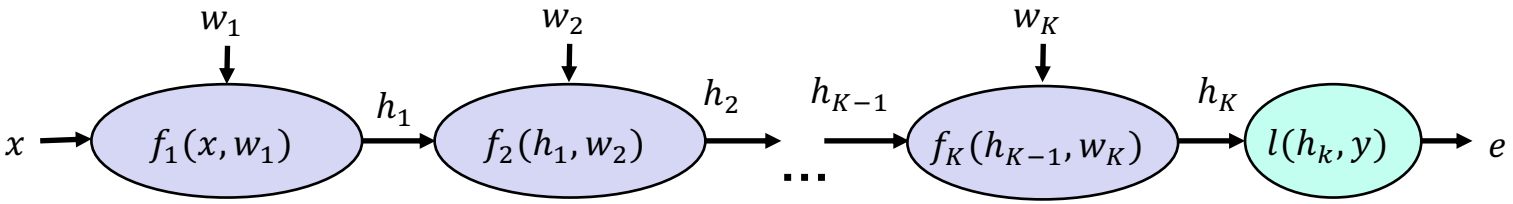


- What is the SGD update for the parameters w_k of the kth layer?

$$w_k \leftarrow w_k - \eta \frac{\partial e}{\partial w_k}$$

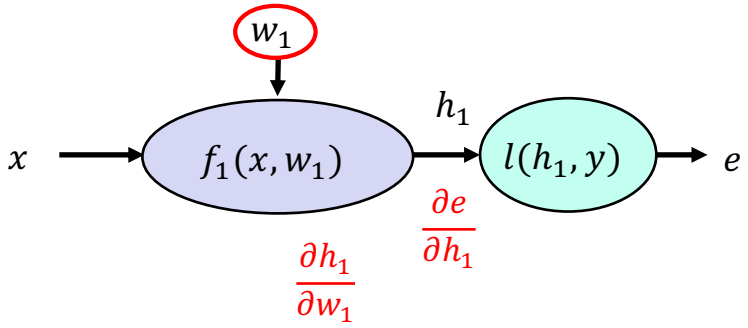
- To train the network, we need to find the gradient of the error w.r.t. the parameters of each layer, $\frac{\partial e}{\partial w_k}$

Computation graph



Chain rule

Let's start with $k = 1$



$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) =$$

Example: $e = (y - w_1^T x)^2$

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$e = l(h_1, y) = (y - h_1)^2$$

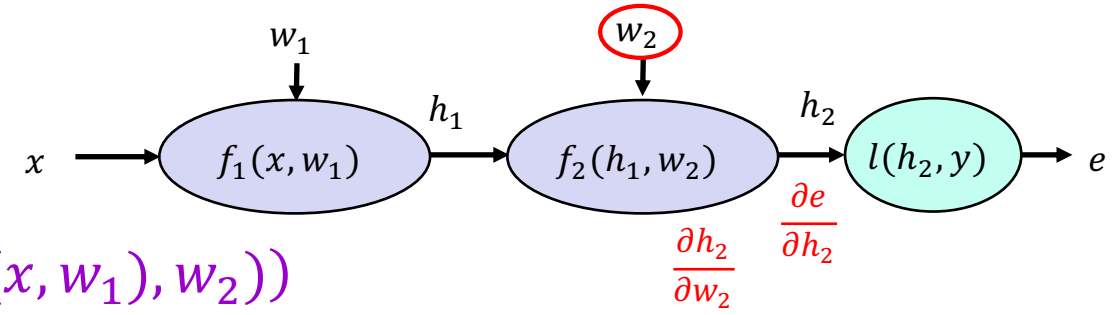
$$\frac{\partial h_1}{\partial w_1} =$$

$$\frac{\partial e}{\partial h_1} =$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Chain rule

$k = 2$

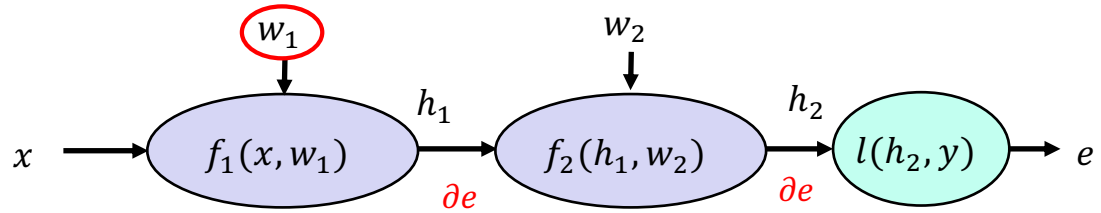


$e = l(f_2(f_1(x, w_1), w_2))$

$$\frac{\partial e}{\partial w_2} =$$

Chain rule

$k = 2$



$e = l(f_2(f_1(x, w_1), w_2))$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2}\frac{\partial h_2}{\partial w_2}$$

Example: $e = -\log\left(\sigma(w_1^T x)\right)$ (assume $y = 1$)

$h_1 = f_1(x, w_1) = w_1^T x$

$h_2 = f_2(h_1) = \sigma(h_1)$

$e = l(h_2, 1) = -\log(h_2)$

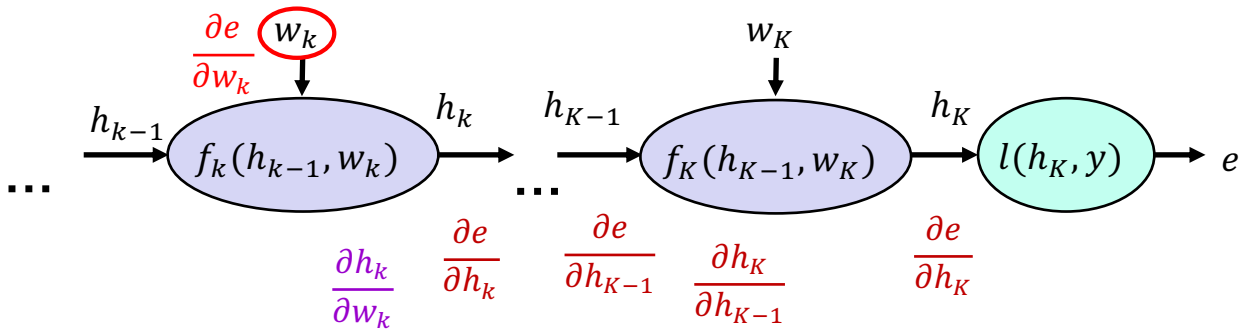
$\frac{\partial h_1}{\partial w_1} =$

$\frac{\partial h_2}{\partial h_1} =$

$\frac{\partial e}{\partial h_2} =$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial w_1} =$$

Chain rule



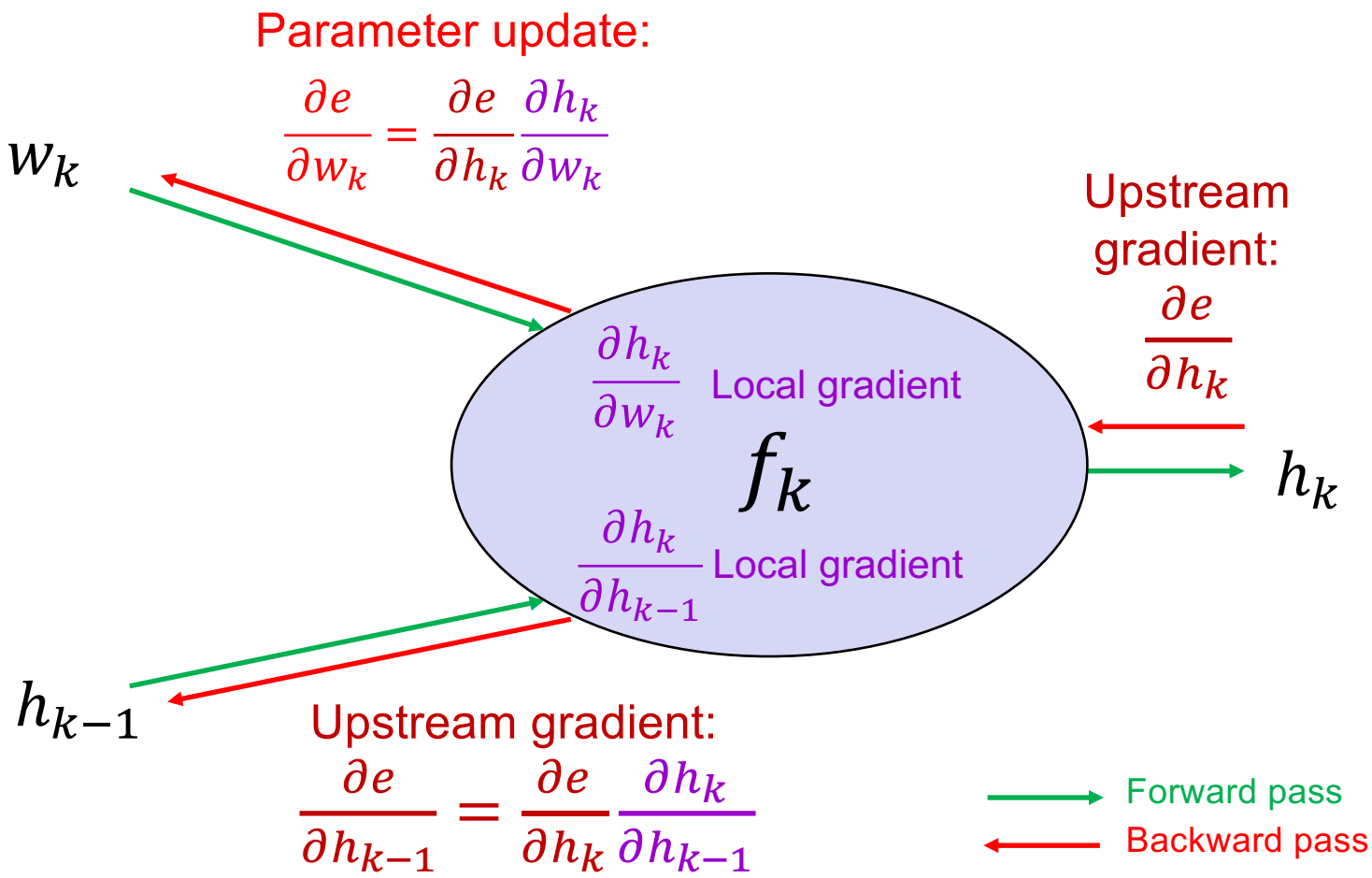
General case:

$$\frac{\partial e}{\partial w_k} = \boxed{\frac{\partial e}{\partial h_K}\ \frac{\partial h_K}{\partial h_{K-1}}\ \cdots\ \frac{\partial h_{k+1}}{\partial h_k}}\ \frac{\partial h_k}{\partial w_k}$$

Upstream gradient $\frac{\partial e}{\partial h_k}$

Local gradient

Backpropagation summary



What about more general computation graphs?

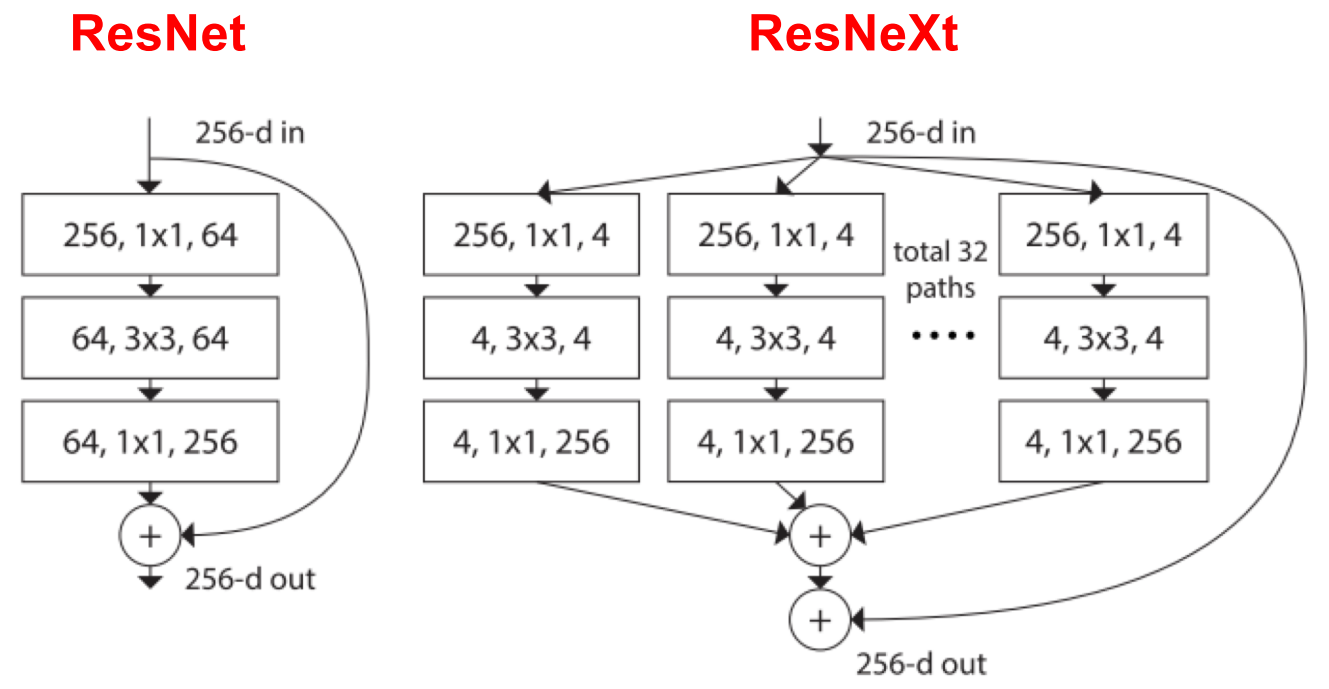
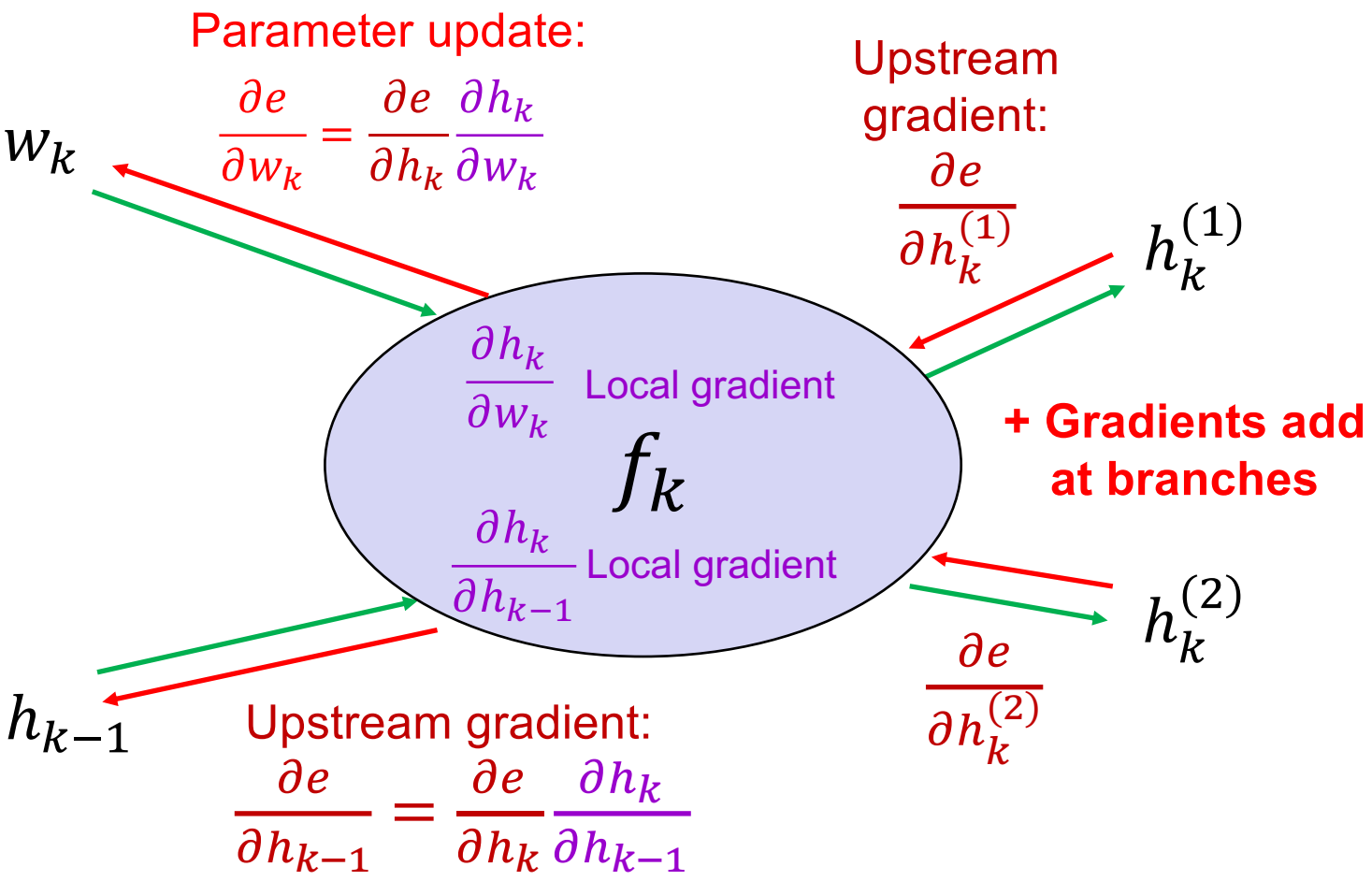


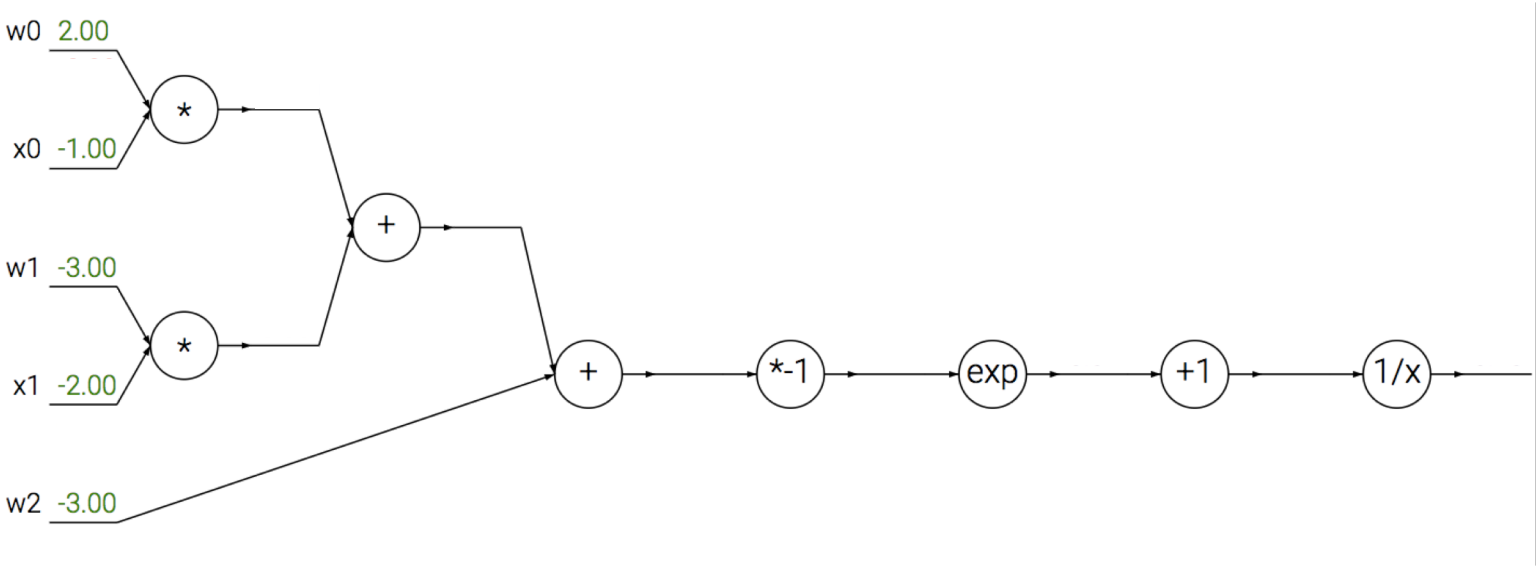
Figure source

What about more general computation graphs?



A detailed example

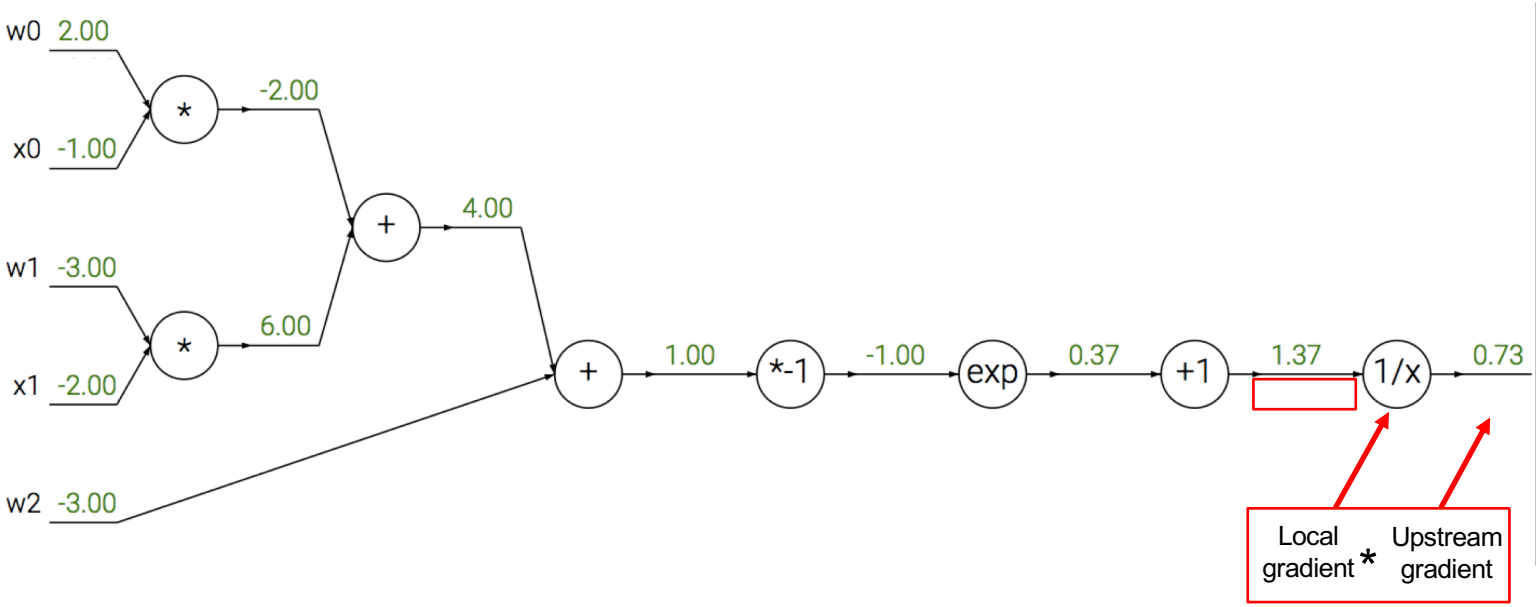
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



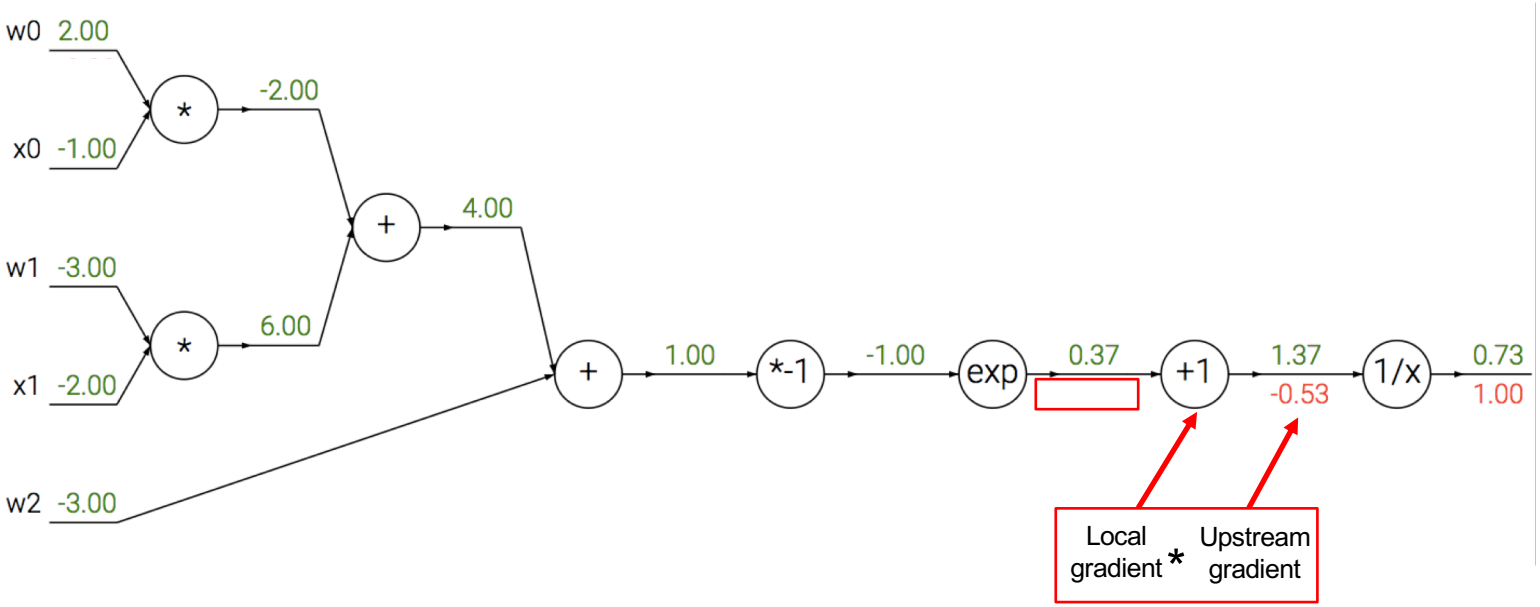
$$(1/x)' = -1/x^2$$

$$-\frac{1}{1.37^2} * 1 = -0.53$$

Source: Stanford 231n

A detailed example

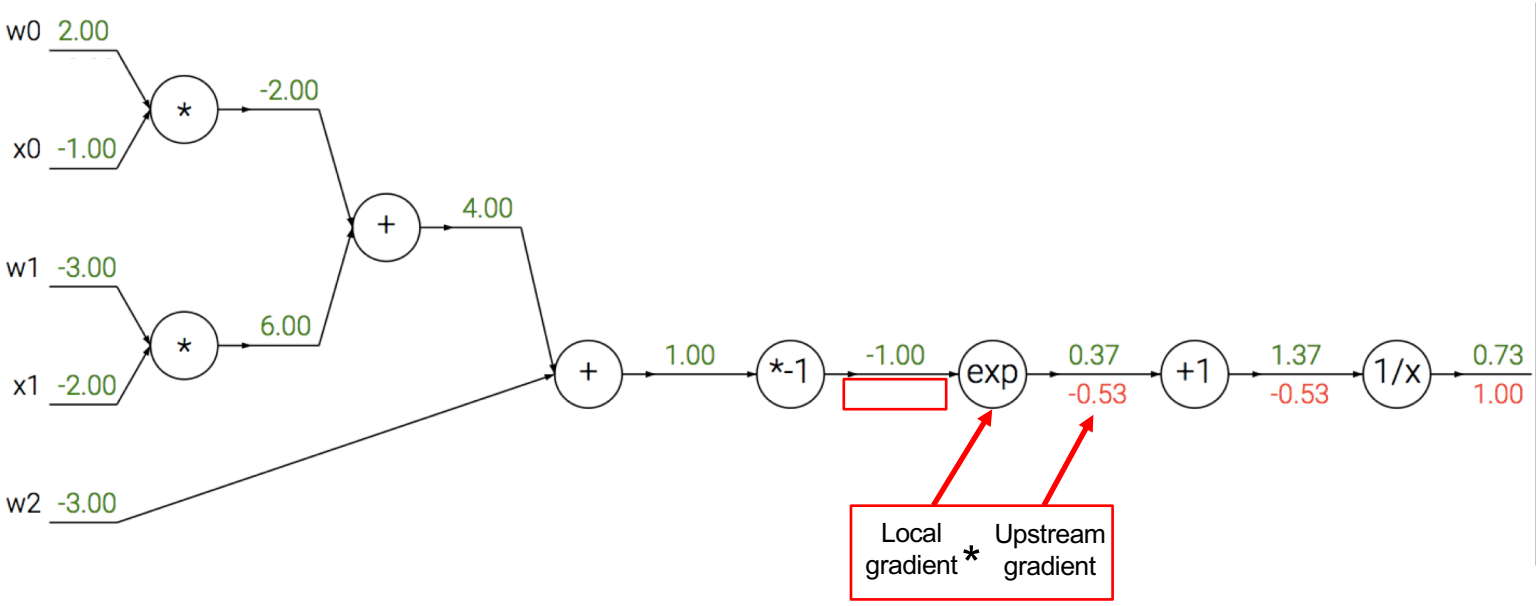
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$

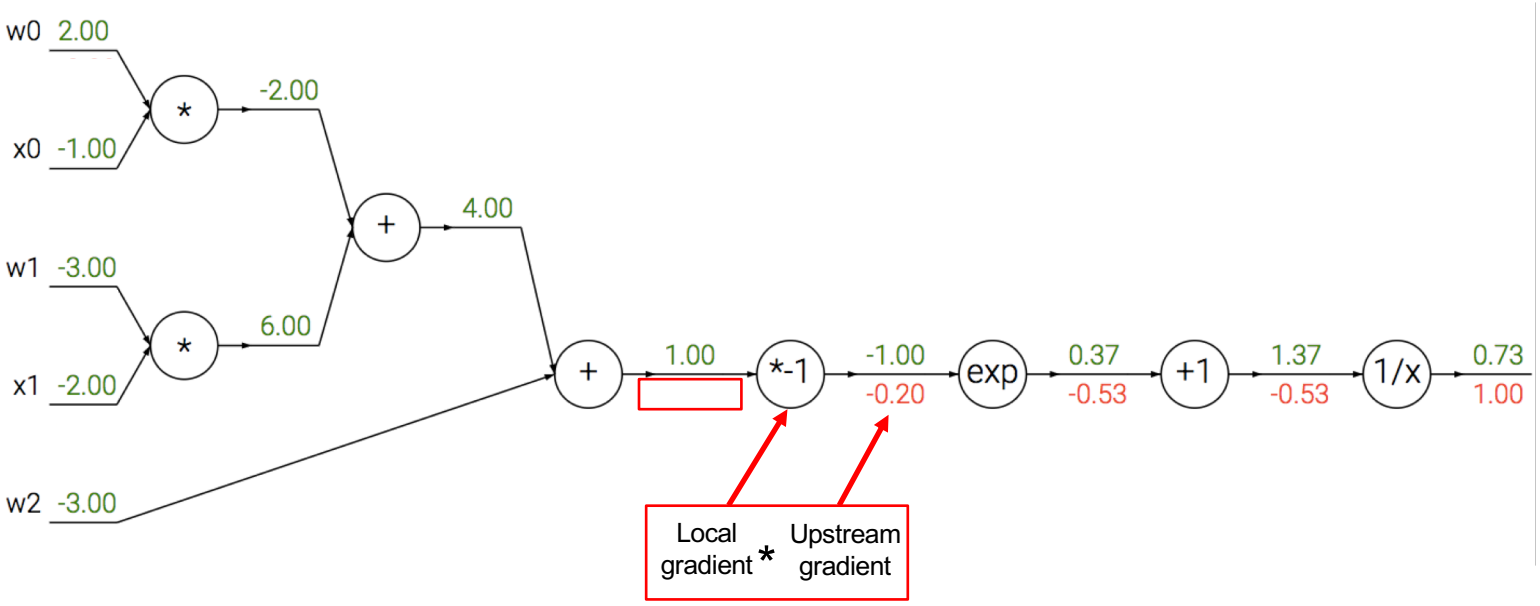


$$\exp(-1) * (-0.53) = -0.20$$

Source: Stanford 231n

A detailed example

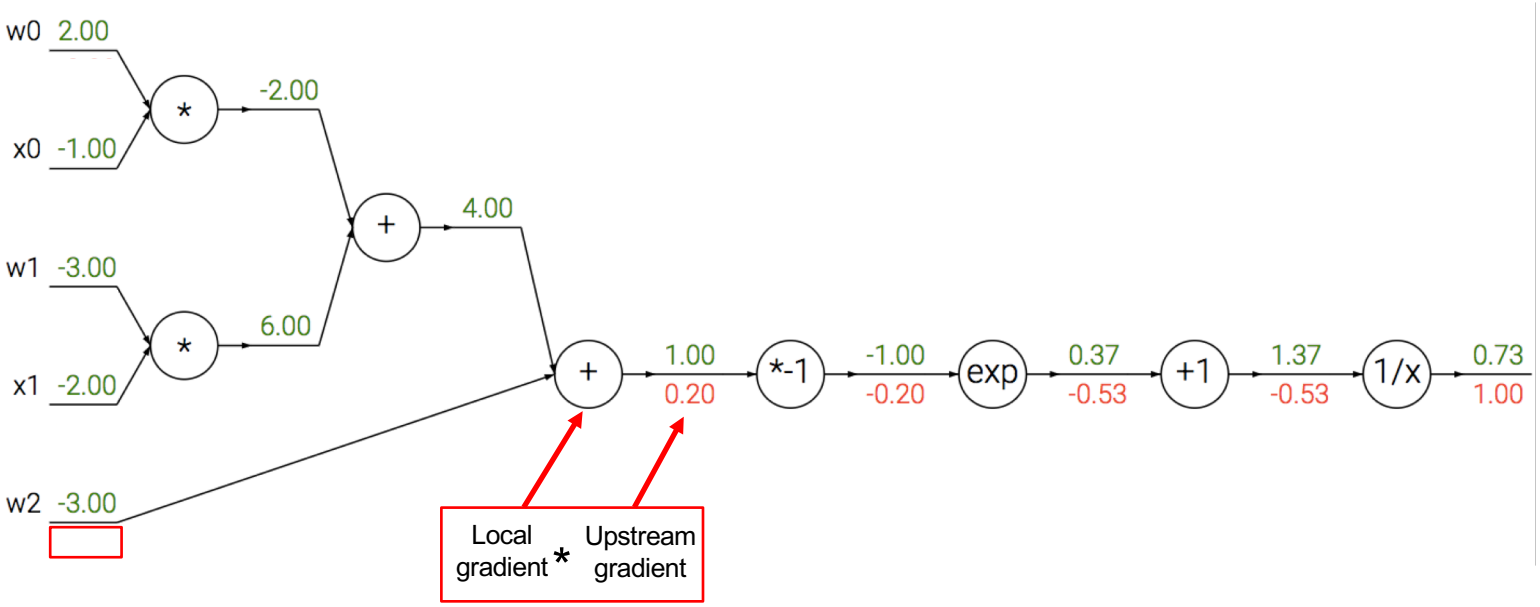
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

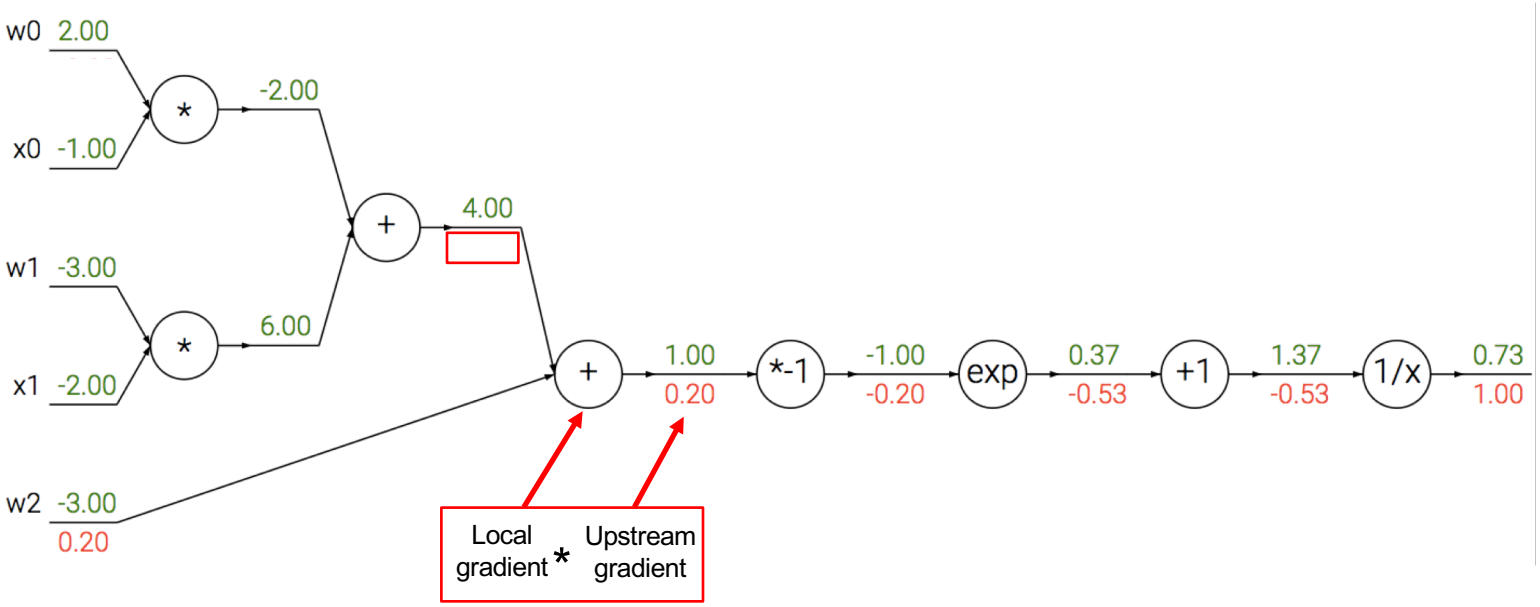
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

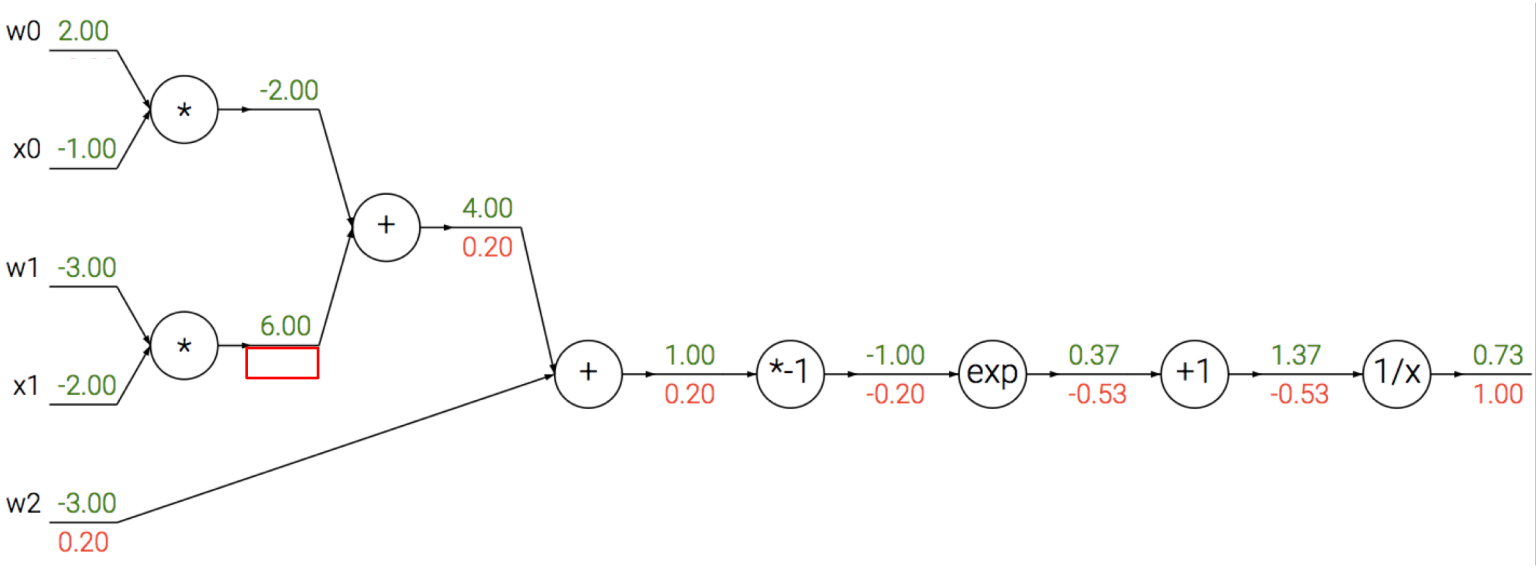
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

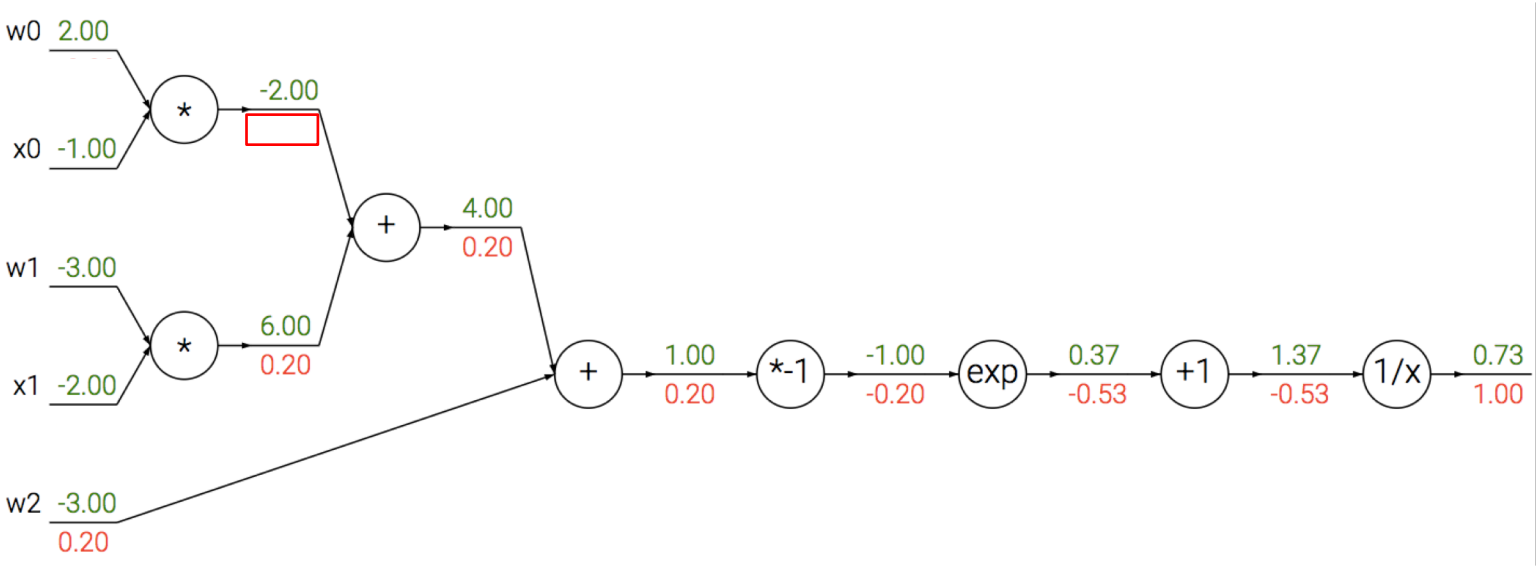
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

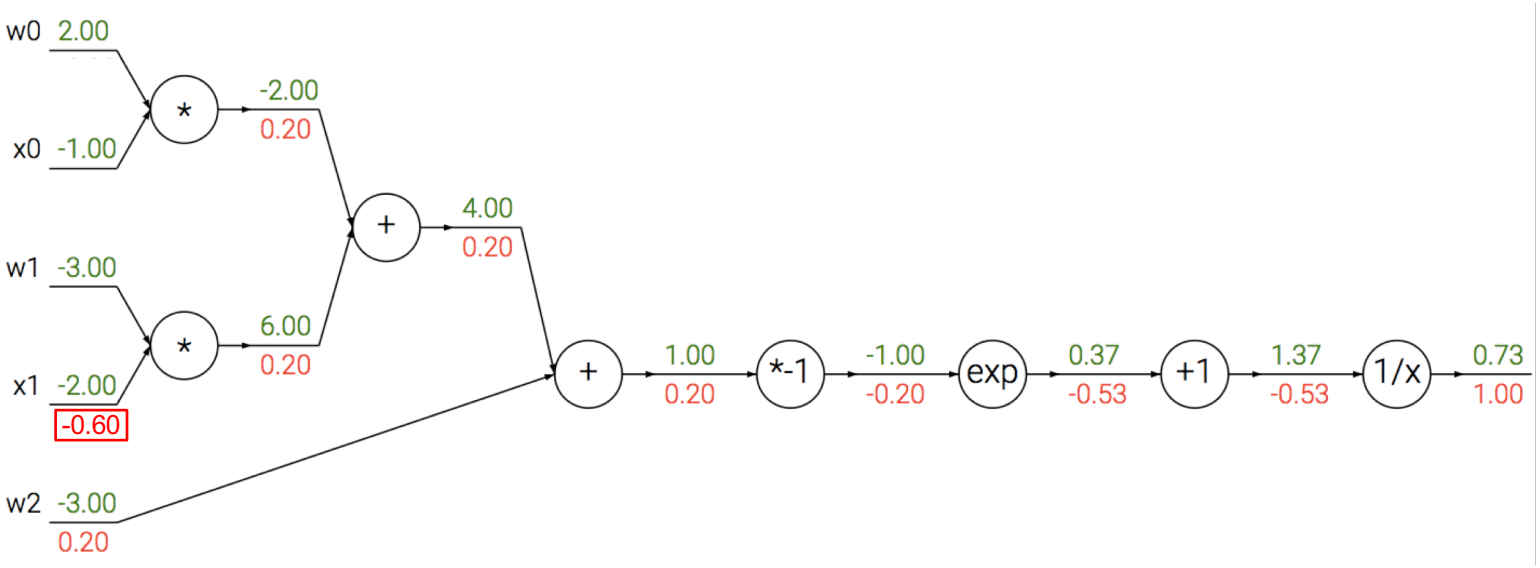
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

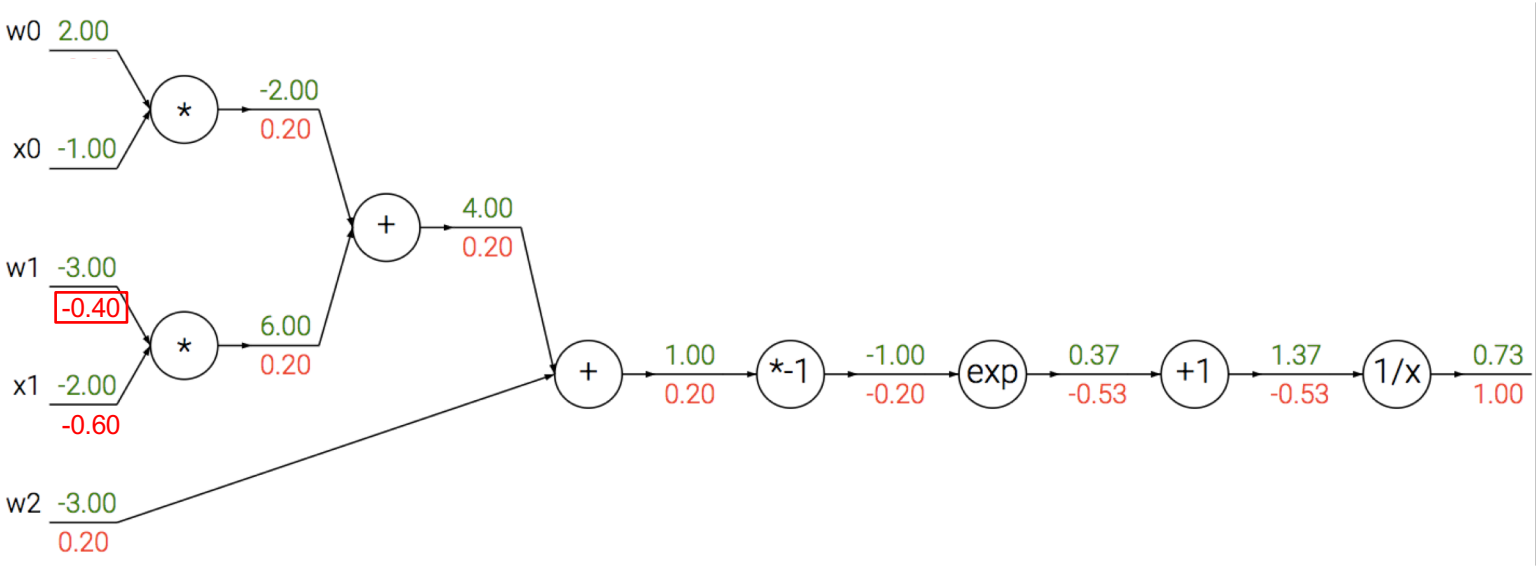
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

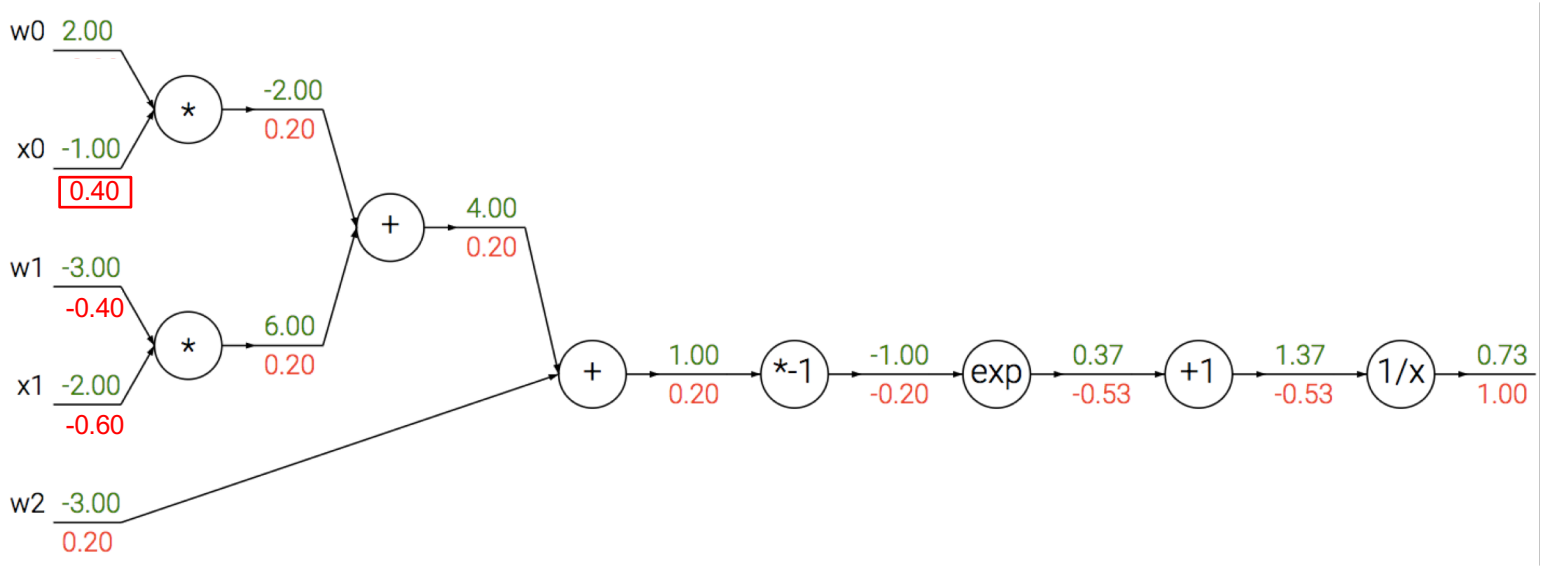
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

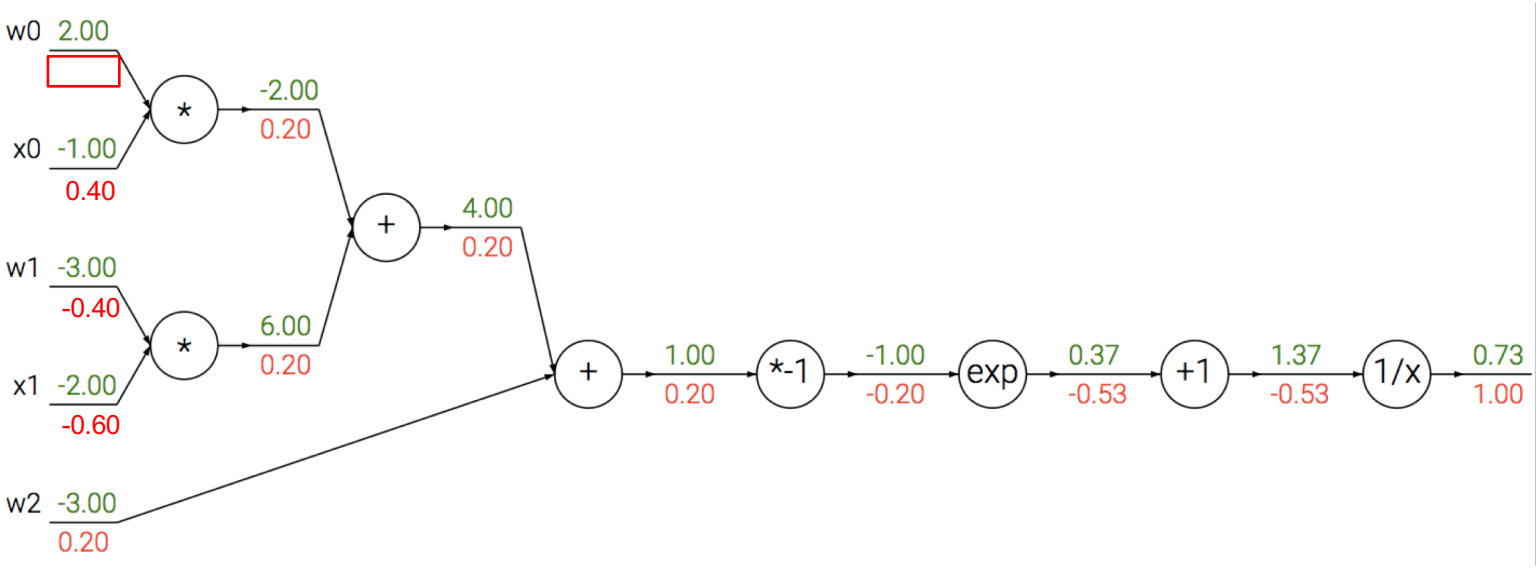
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

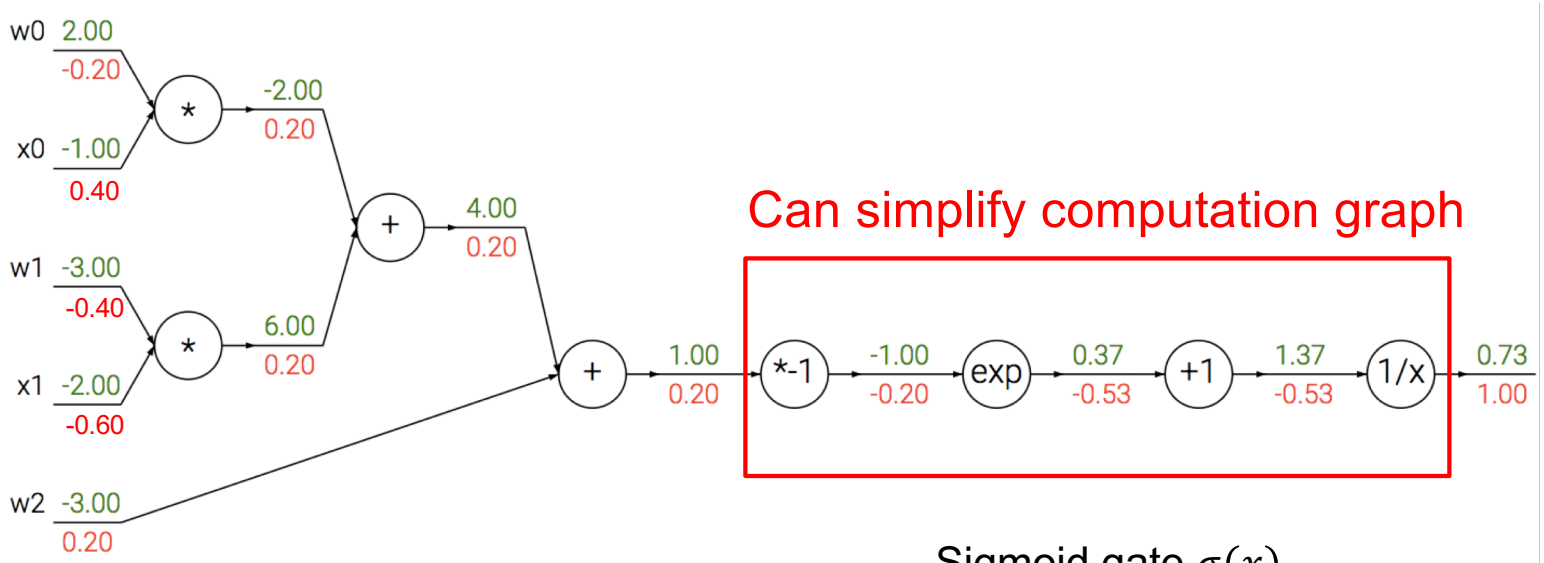
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



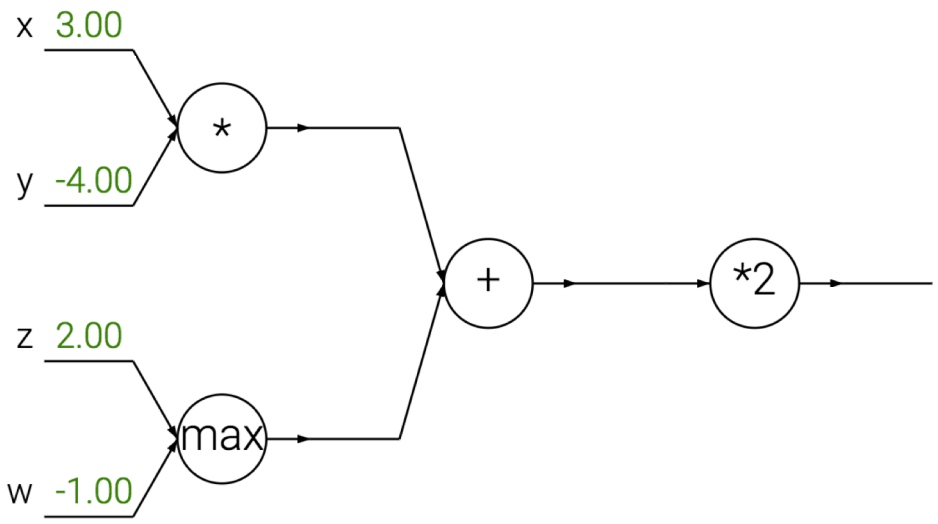
Sigmoid gate $\sigma(x)$

$$\sigma'(x) = \sigma(x)\big(1-\sigma(x)\big)$$

$$\sigma(1)\big(1-\sigma(1)\big) = 0.73 * (1-0.73) = 0.20$$

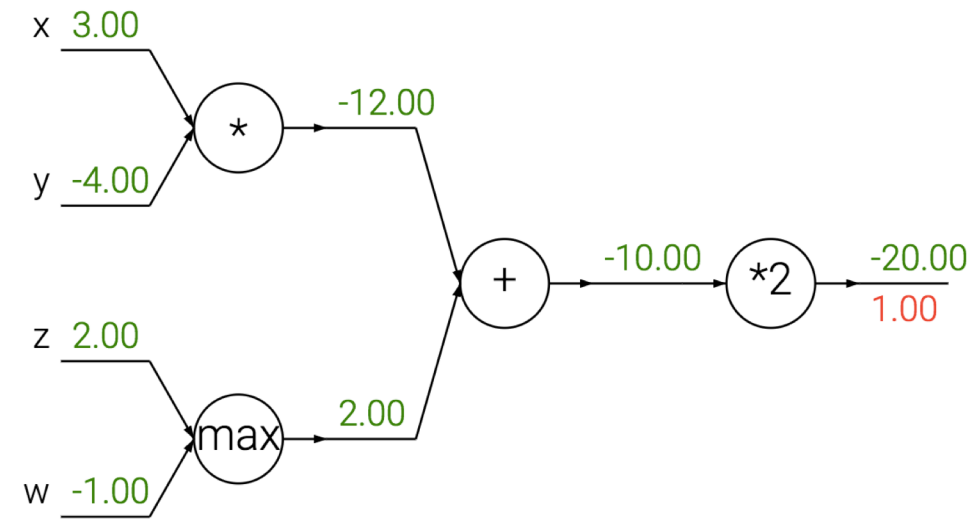
Source: Stanford 231n

Patterns in gradient flow



Source: Stanford 231n

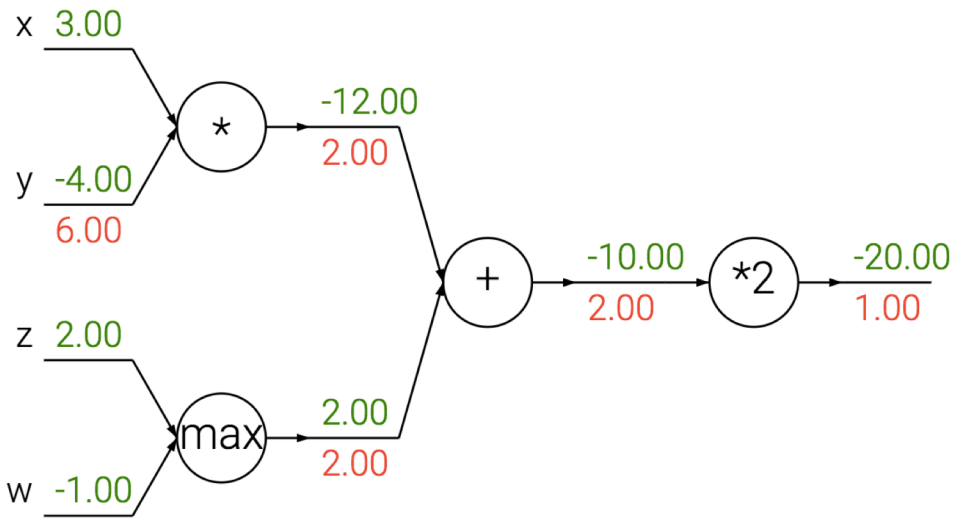
Patterns in gradient flow



Add gate: “gradient distributor”

Source: Stanford 231n

Patterns in gradient flow

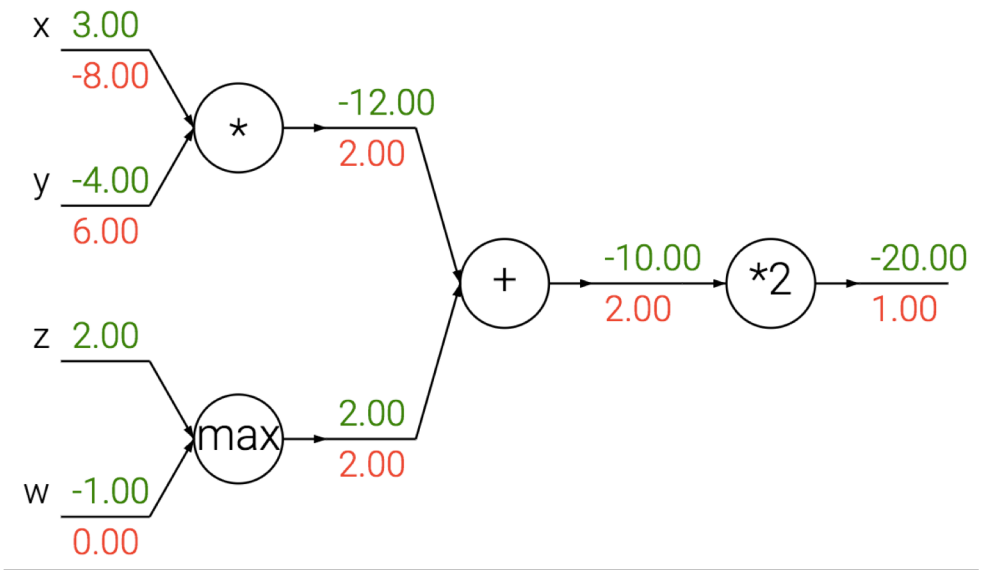


Add gate: “gradient distributor”

Multiply gate: “gradient switcher”

Source: Stanford 231n

Patterns in gradient flow



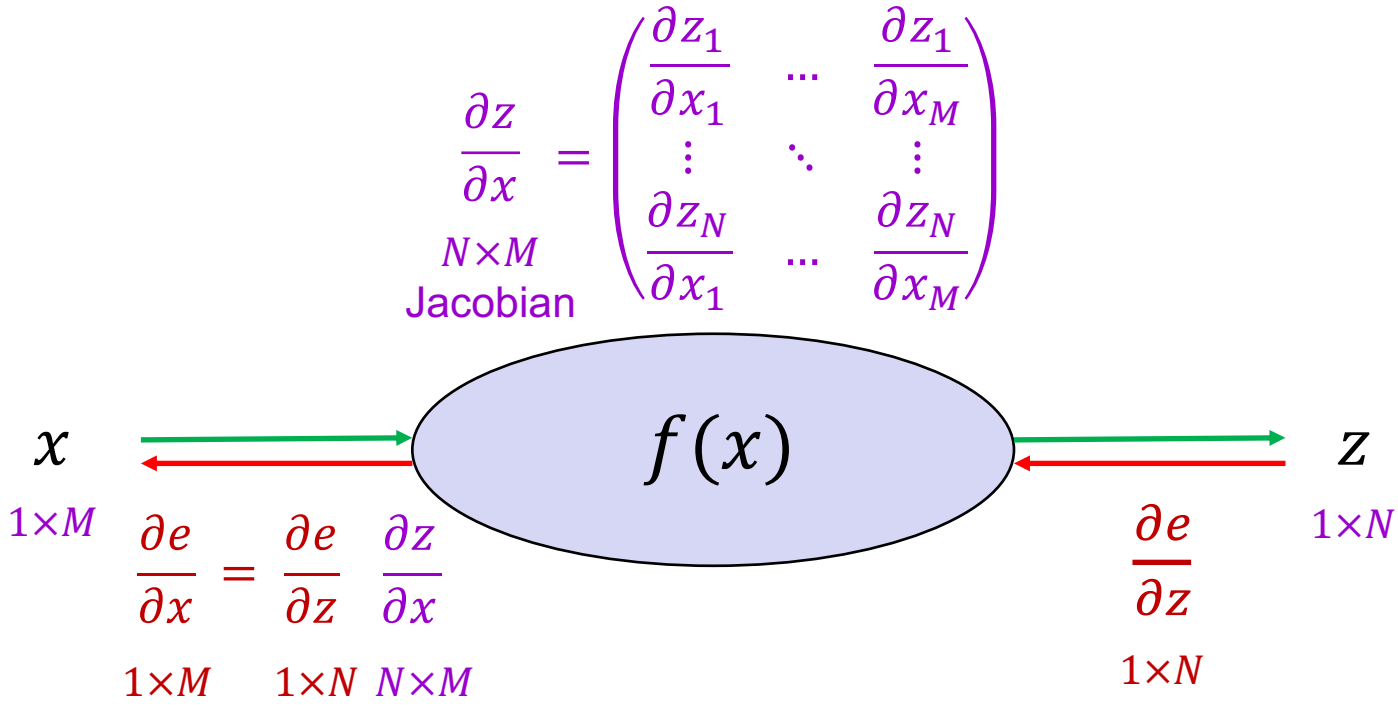
Add gate: “gradient distributor”

Multiply gate: “gradient switcher”

Max gate: “gradient router”

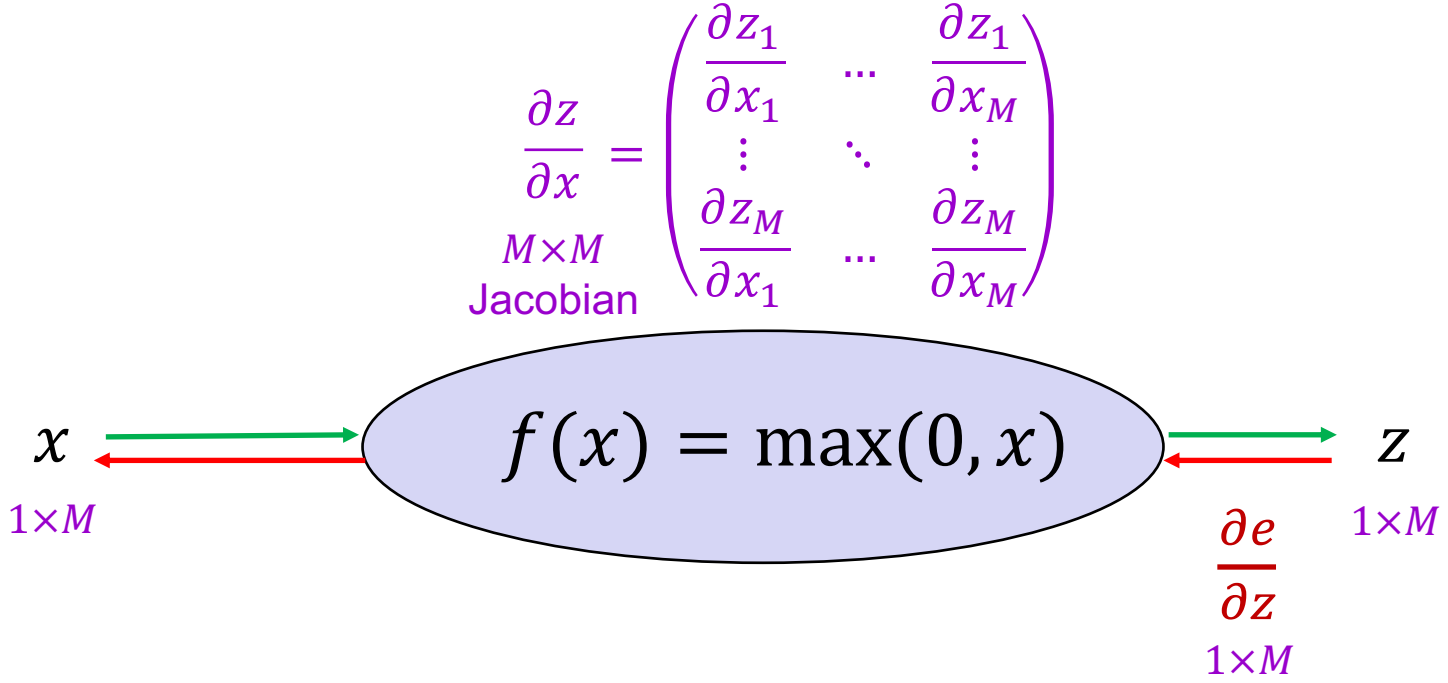
Source: Stanford 231n

Dealing with vectors

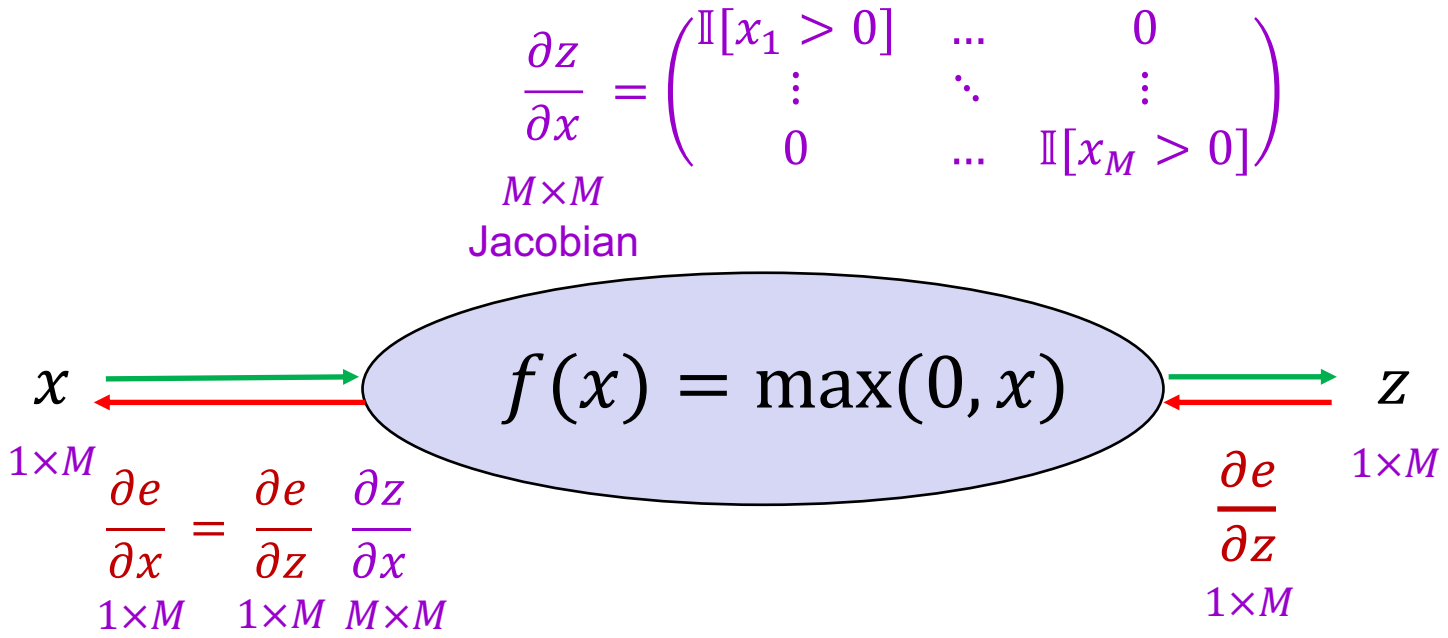


Simple case: Elementwise operation

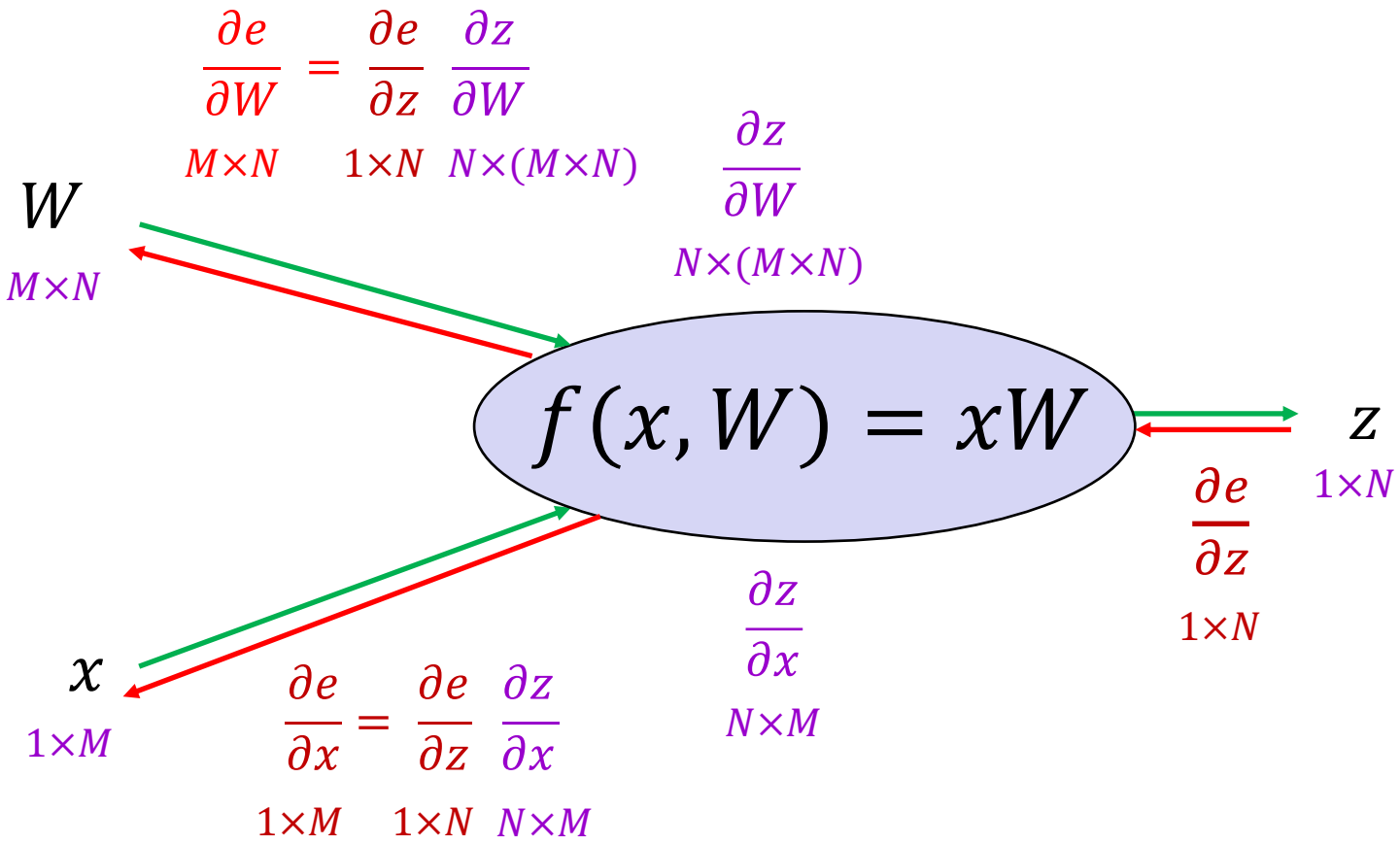
Simple case: Elementwise operation (ReLU layer)



Simple case: Elementwise operation (ReLU layer)



Matrix-vector multiplication (linear layer)



Matrix-vector multiplication (linear layer)

$$(z_1 \quad \dots \quad z_N) = (x_1 \quad \dots \quad x_M) \begin{pmatrix} W_{11} & \dots & W_{1N} \\ \vdots & \ddots & \vdots \\ W_{M1} & \dots & W_{MN} \end{pmatrix} \qquad z_j = \sum_{i=1}^{M} x_i W_{ij}$$

Want: $\underset{1\times M}{\frac{\partial e}{\partial x}} = \underset{1\times N}{\frac{\partial e}{\partial z}} \underset{N\times M}{\boxed{\frac{\partial z}{\partial x}}}$

$$\frac{\partial z_j}{\partial x_i} =$$

jth row, ith column of Jacobian

$$\frac{\partial z}{\partial x} = W^T$$

$$\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial e}{\partial z} W^T$$

Matrix-vector multiplication (linear layer)

$$(z_1 \quad \dots \quad z_N) = (x_1 \quad \dots \quad x_M) \begin{pmatrix} W_{11} & \dots & W_{1N} \\ \vdots & \ddots & \vdots \\ W_{M1} & \dots & W_{MN} \end{pmatrix} \qquad z_j = \sum_{i=1}^{M} x_i W_{ij}$$

Want: $\underset{M\times N}{\frac{\partial e}{\partial W}} = \underset{1\times N}{\frac{\partial e}{\partial z}} \underset{N\times(M\times N)}{\boxed{\frac{\partial z}{\partial W}}}$

$$\frac{\partial z_k}{\partial W_{ij}} =$$

z_k depends only on kth column of W

$$\frac{\partial e}{\partial W_{ij}} =$$

$$\frac{\partial e}{\partial W} = x^T \frac{\partial e}{\partial z}$$

General tips

- Derive error signal (upstream gradient) directly, avoid explicit computation of huge local derivatives
- Write out expression for a single element of the Jacobian, then deduce the overall formula
- Keep consistent indexing conventions, order of operations
- Use dimension analysis

- **For further reading:**
  - Lecture 4 of Stanford 231n and associated links in the syllabus
  - Yes you should understand backprop by Andrej Karpathy