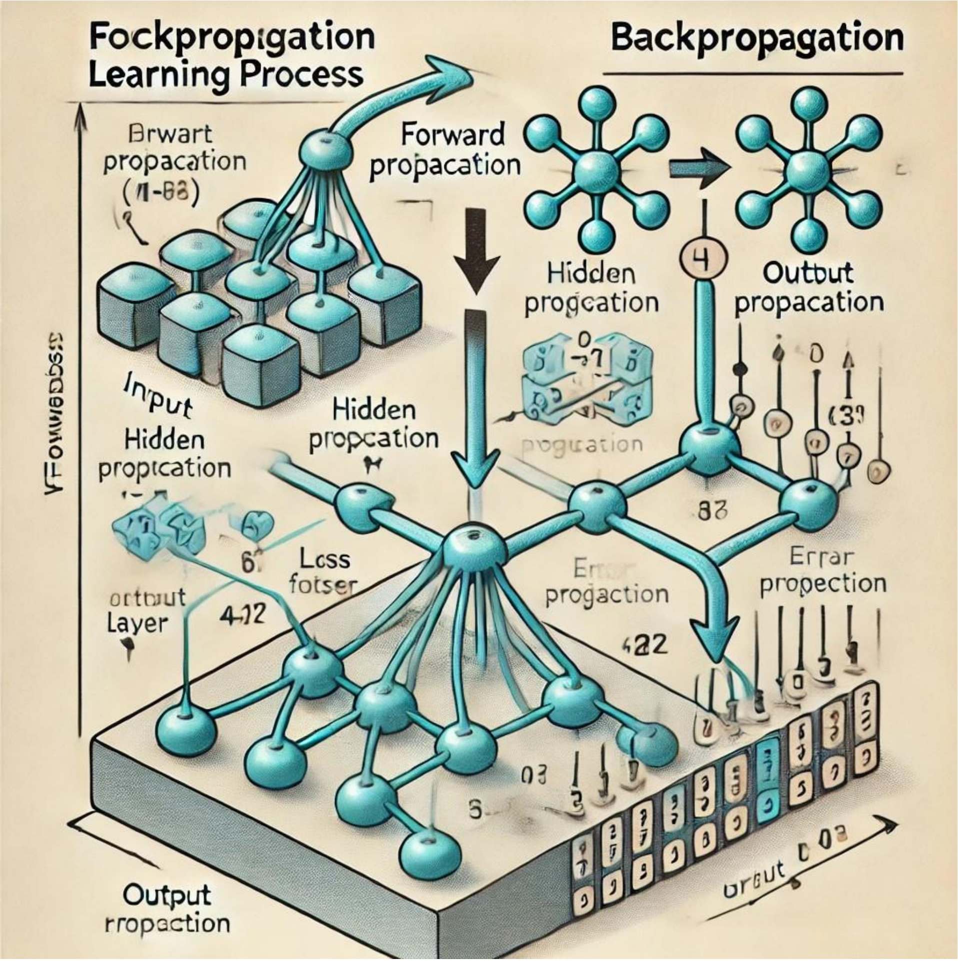
Back Propagation



AI604 Deep Learning for Computer Vision

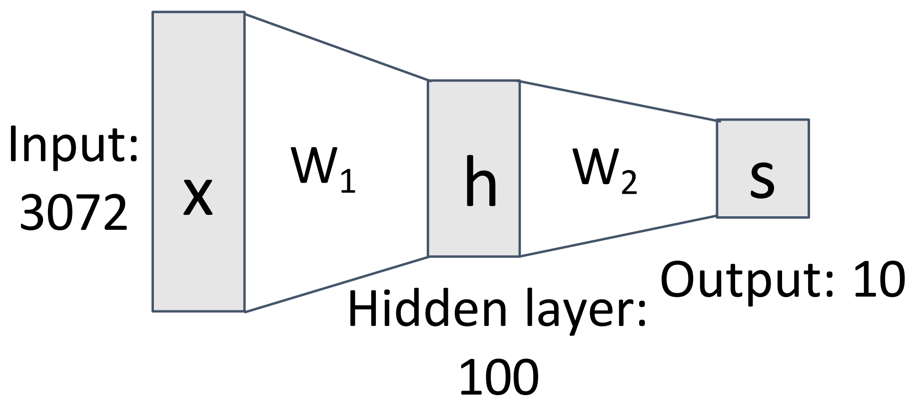
Prof. Hyunjung Shim

Slide credit: Justin Johnson, Fei-Fei Li, Ehsan Adeli

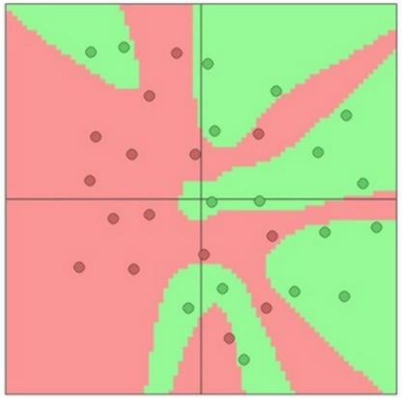
Recap: Classification/Loss/Optimizer

From linear classifiers to fully-connected networks

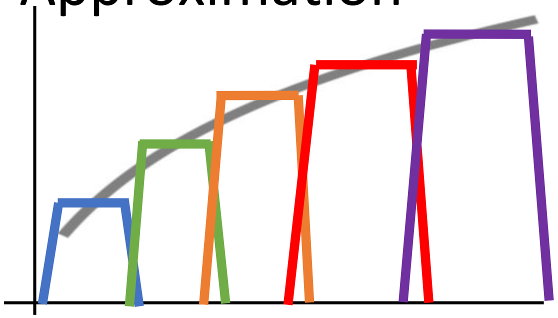
$$f = W_2 \max(0, W_1 x)$$



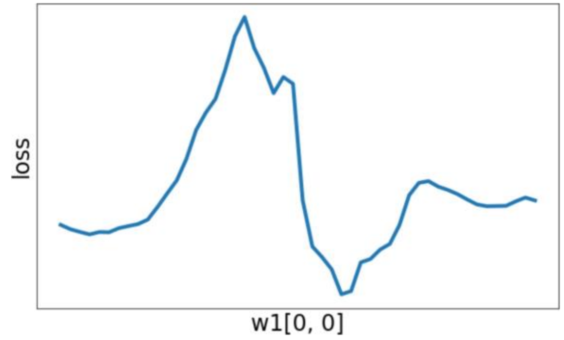
Space Warping



Universal Approximation



Nonconvex



(Bad) Idea: Derive $\nabla_W L$ on paper

$$s = f(x; W) = Wx$$

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

$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$

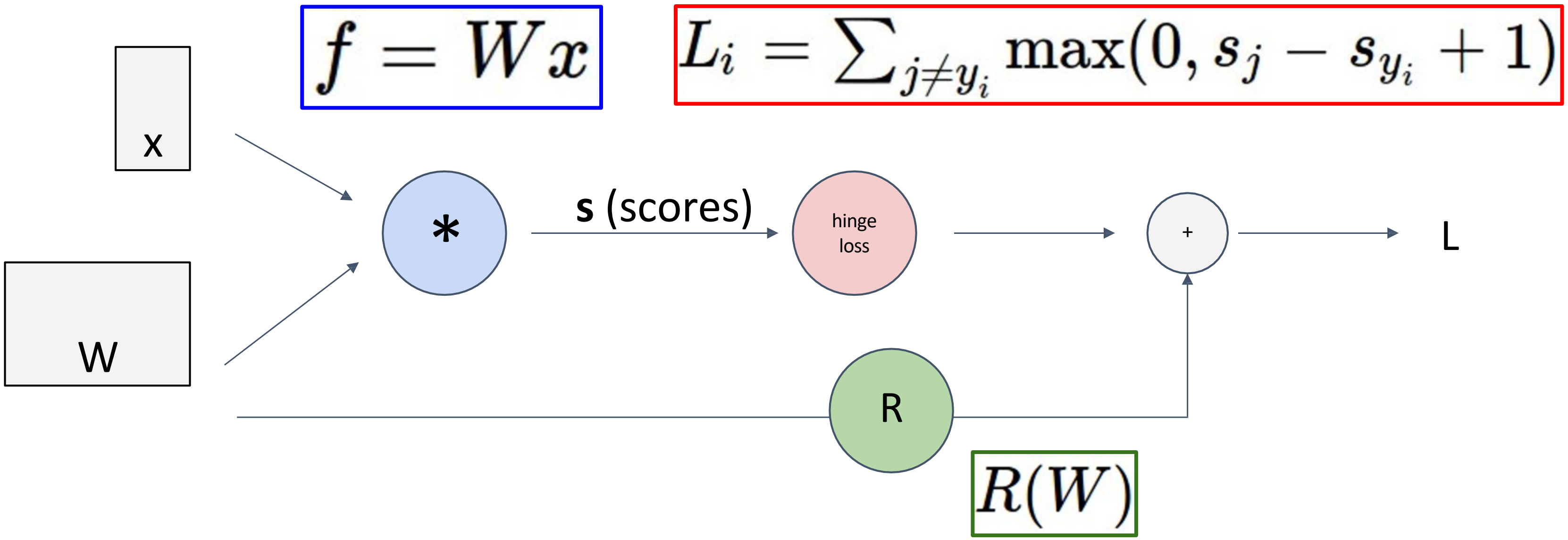
$$\nabla_W L = \nabla_W \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$

Problem: Very tedious: Lots of matrix calculus, need lots of paper

Problem: What if we want to change loss? E.g. use softmax instead of SVM? Need to re-derive from scratch. Not modular!

Problem: Not feasible for very complex models!

Better Idea: Computational Graphs

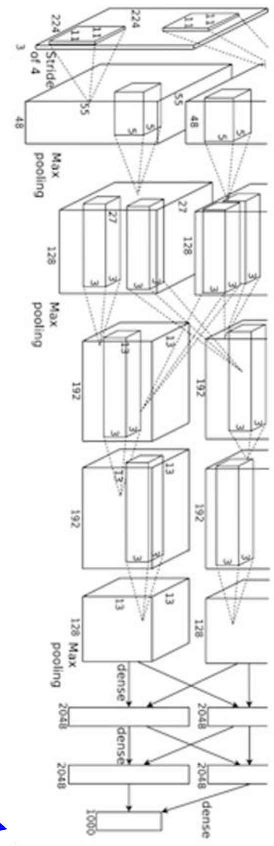


Deep Network (AlexNet)

input image

weights

loss



Neural Turing Machine

input image

loss

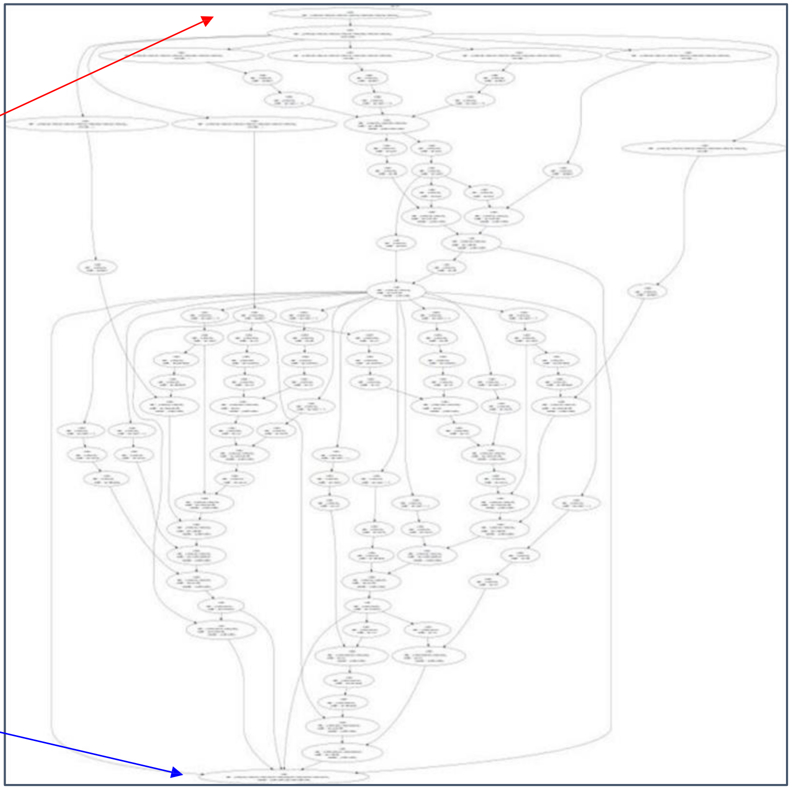


Figure reproduced with permission from a Twitter post by Andrej Karpathy.

Neural Turing Machine

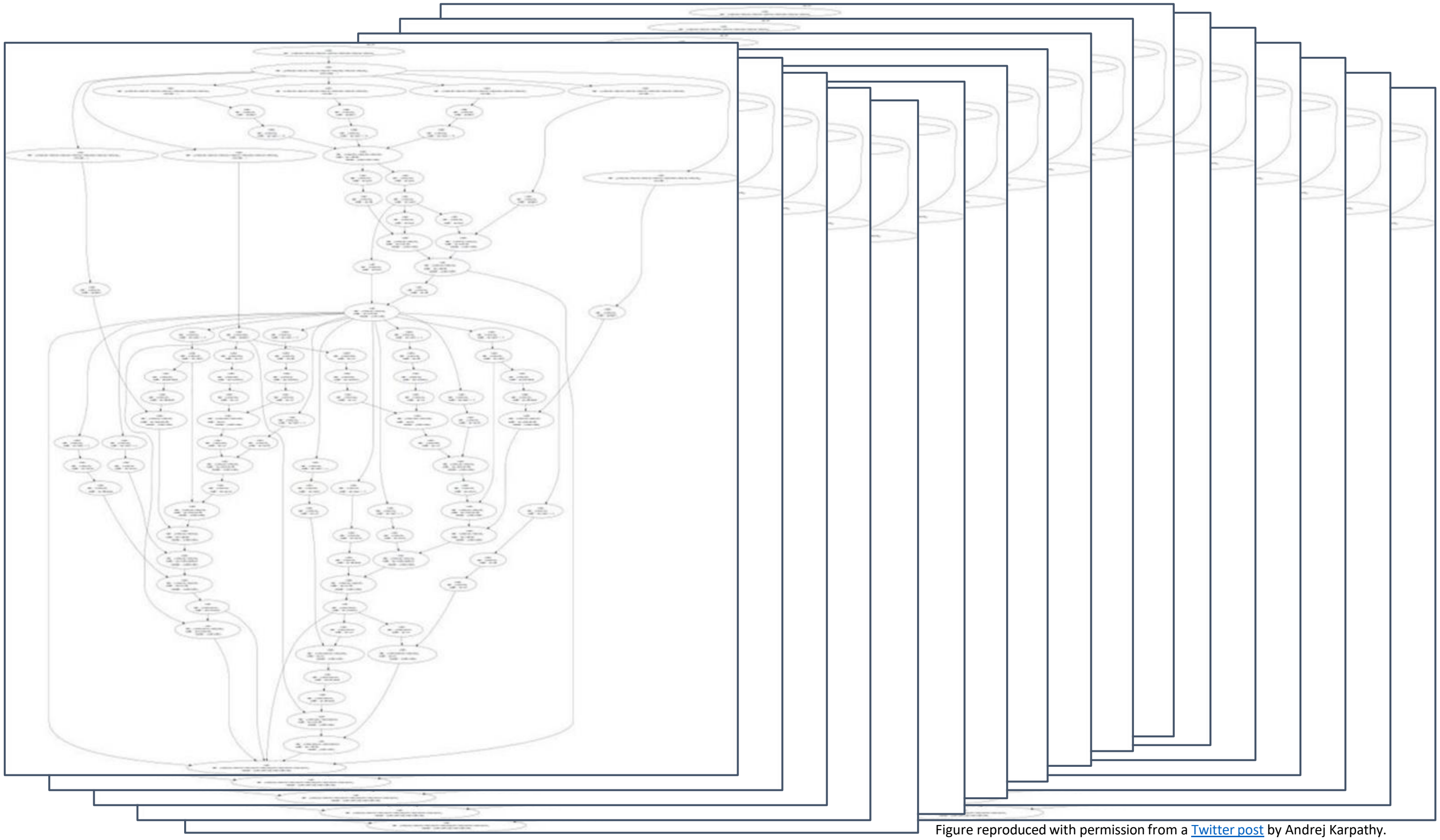
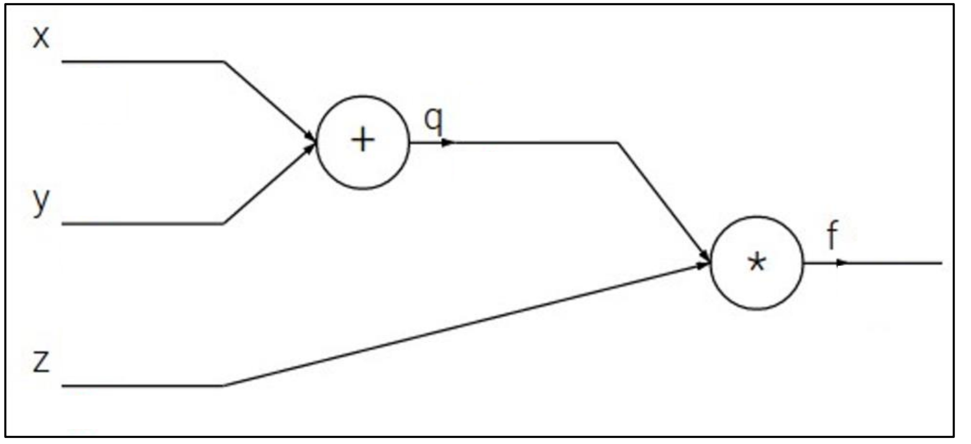


Figure reproduced with permission from a Twitter post by Andrej Karpathy.

Backpropagation: Simple Example

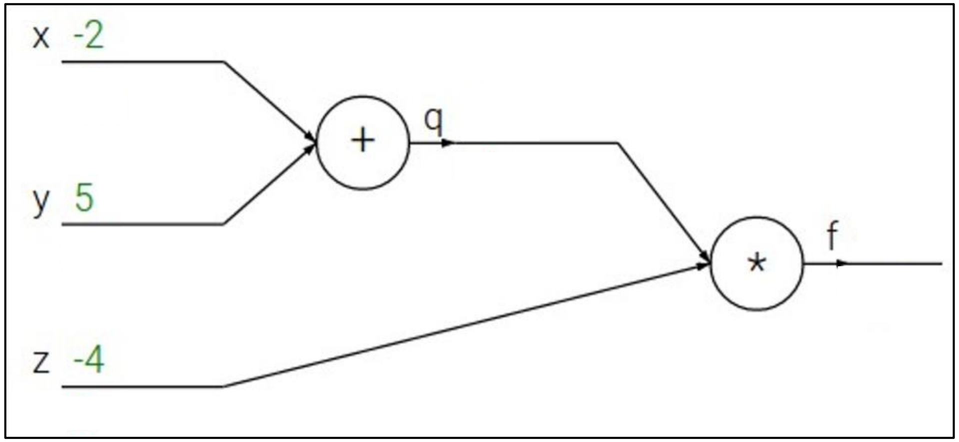
$$f(x, y, z) = (x + y)z$$



Backpropagation: Simple Example

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

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



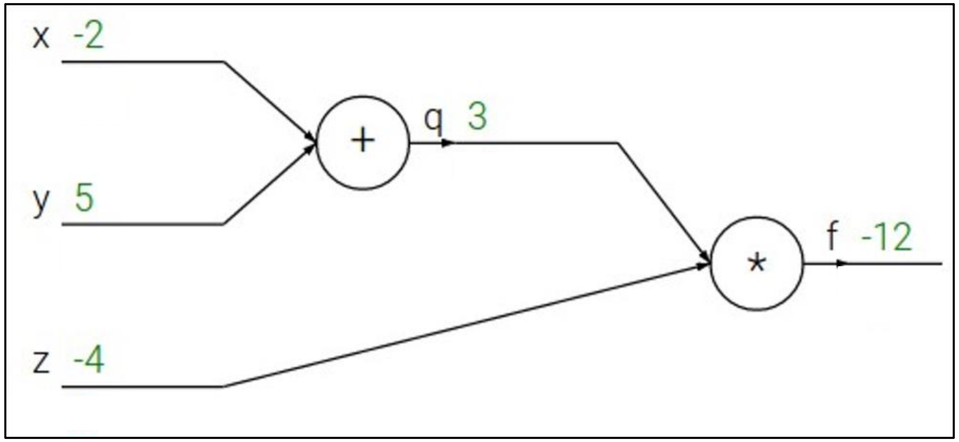
Backpropagation: Simple Example

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

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

1. Forward pass: Compute outputs

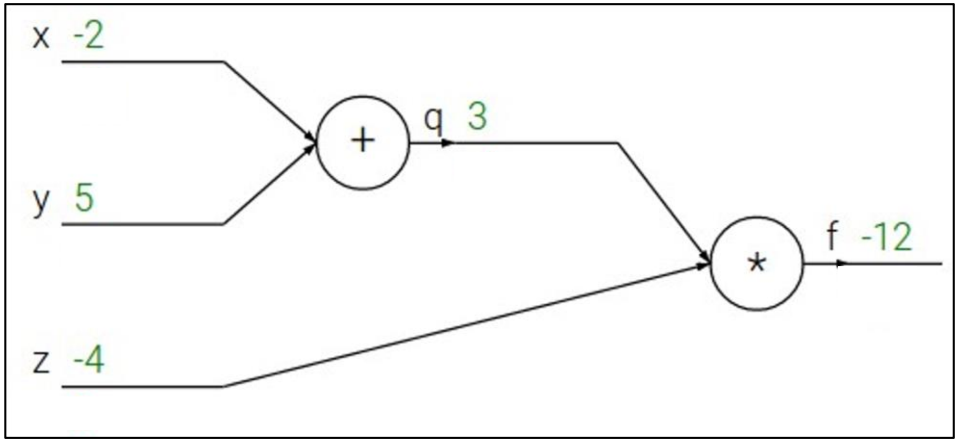
$$q = x + y \qquad f = qz$$



Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

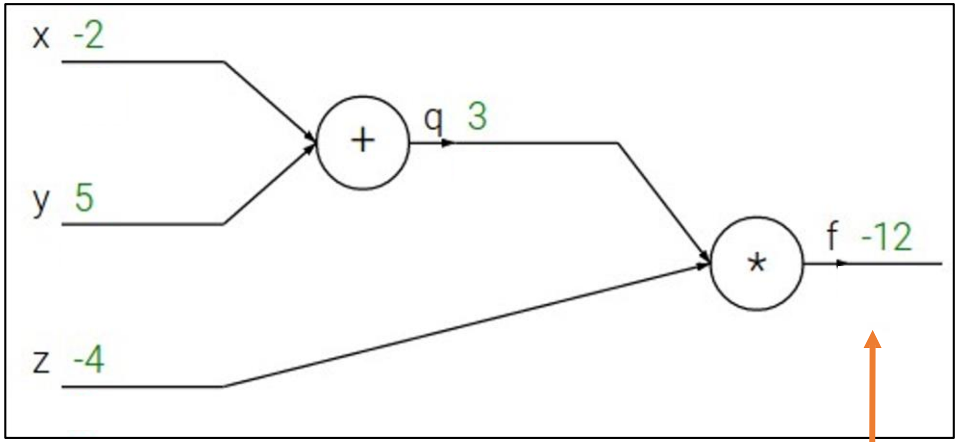
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



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

1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

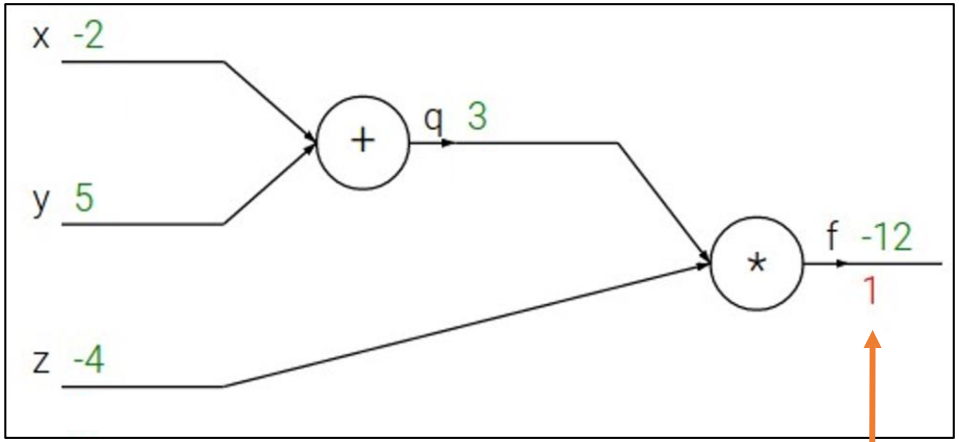
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



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

1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

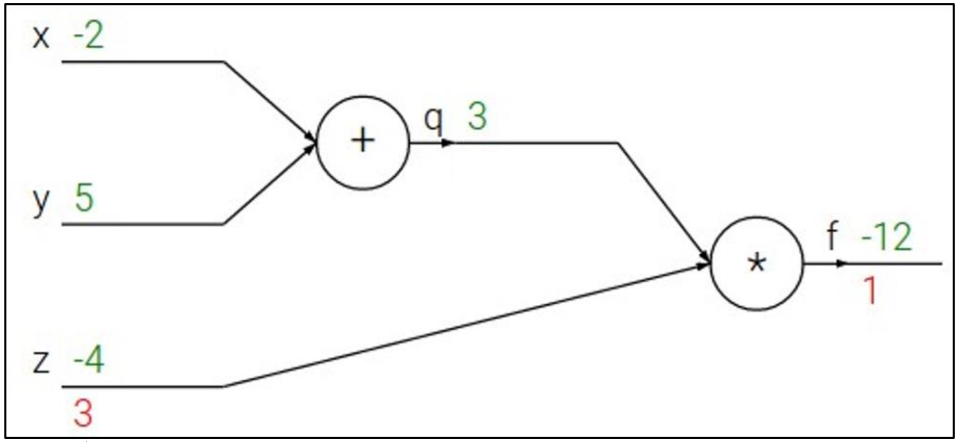
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad \boxed{f = qz}$$

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

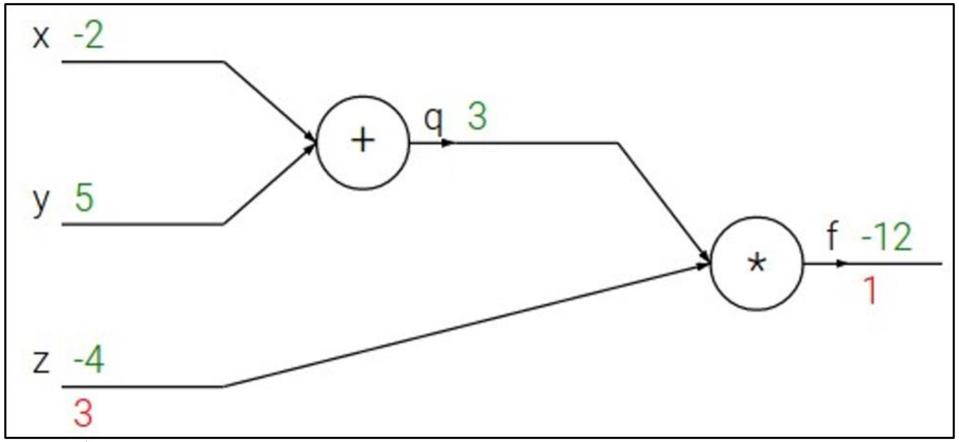
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad \boxed{f = qz}$$

$$\boxed{\frac{\partial f}{\partial z} = q}$$

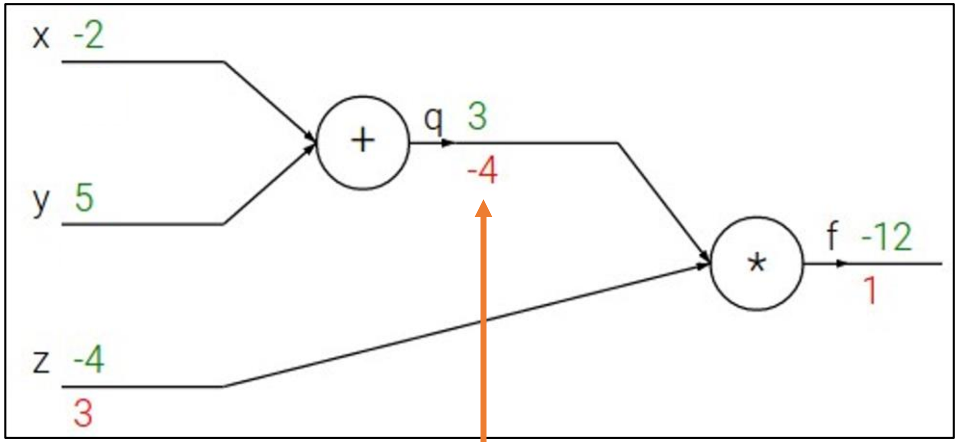
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



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

1. Forward pass: Compute outputs

$$q = x + y \qquad \boxed{f = qz}$$

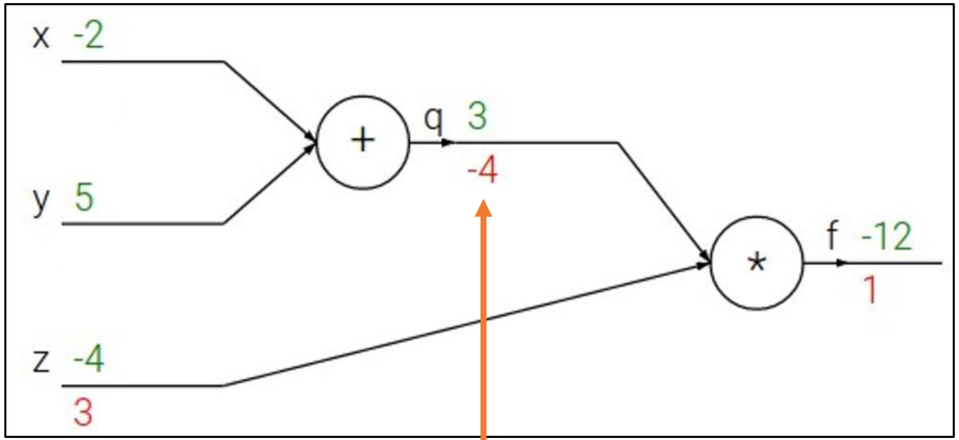
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad \boxed{f = qz}$$

$$\boxed{\frac{\partial f}{\partial q} = z}$$

2 . Backward pass: Compute derivatives

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

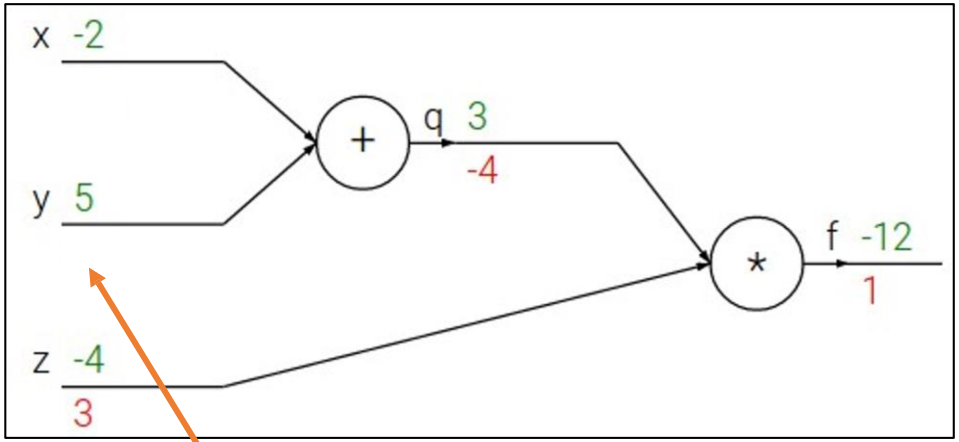
Backpropagation: Simple Example

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

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

1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$



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

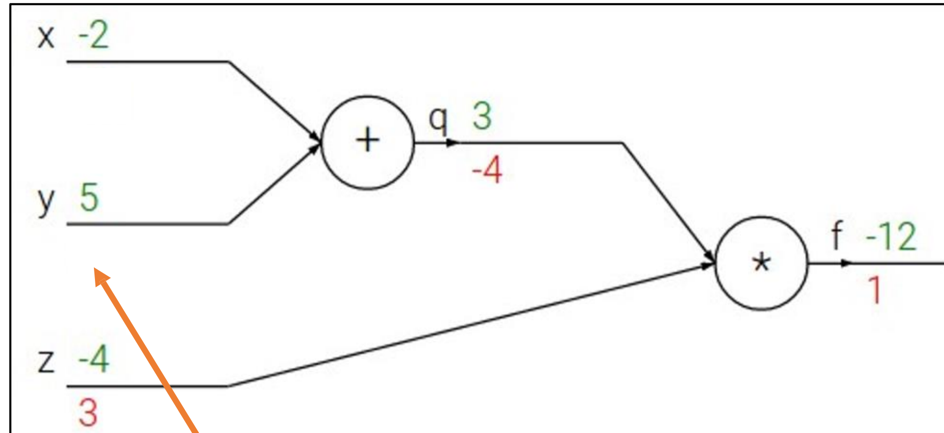
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

Chain Rule

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

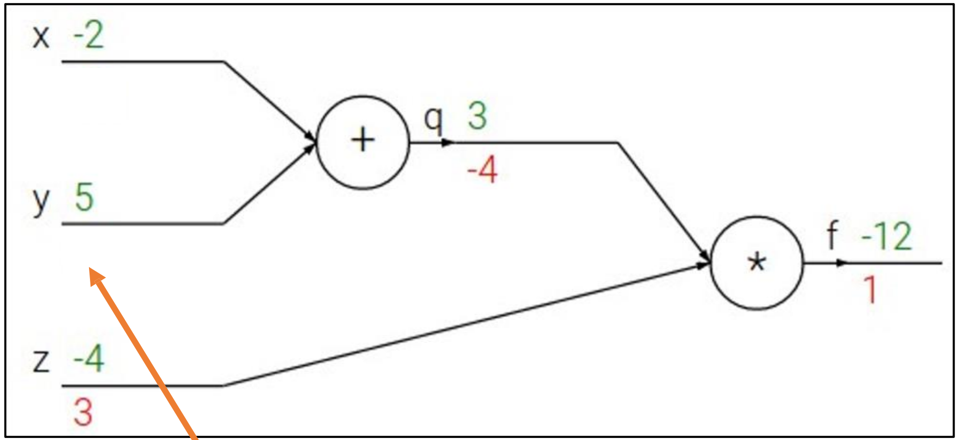
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

2 . Backward pass: Compute derivatives

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

Chain Rule

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

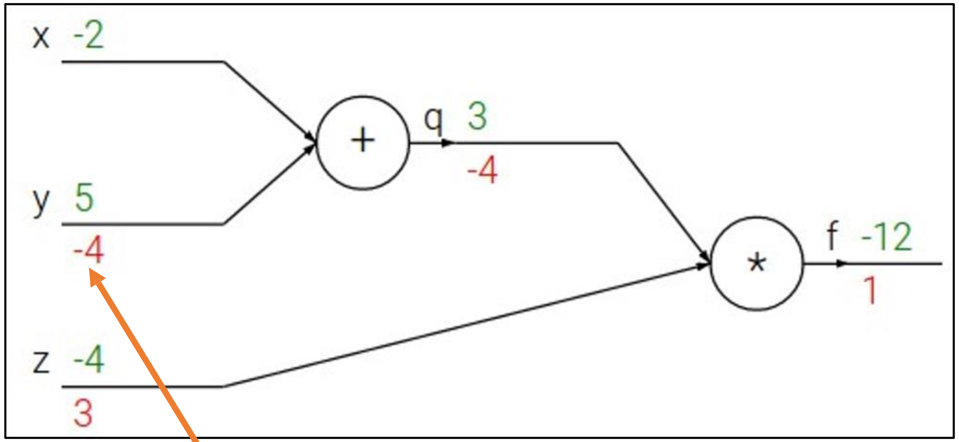
Downstream Gradient **Local Gradient** **Upstream Gradient**

$$\frac{\partial q}{\partial y} = 1$$

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$\boxed{q = x + y} \qquad f = qz$$

Chain Rule

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial q}{\partial y} \frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial y} = 1}$$

Downstream Gradient **Local Gradient** **Upstream Gradient**

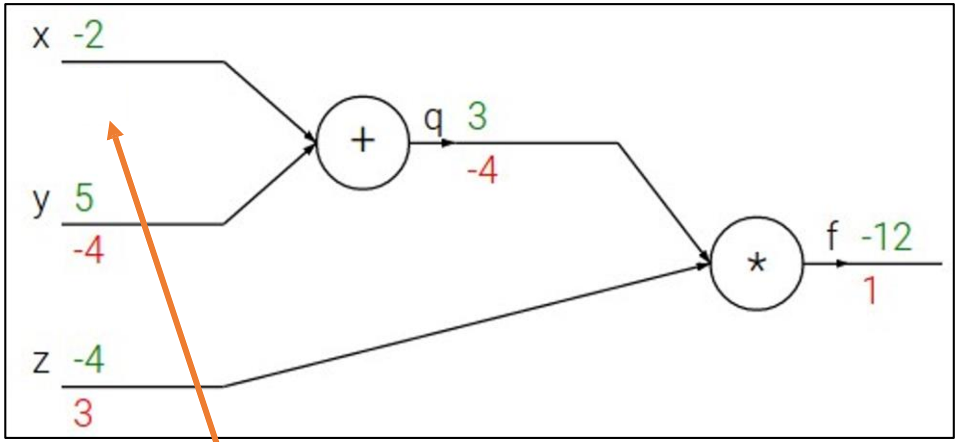
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$\boxed{q = x + y} \qquad f = qz$$

Chain Rule

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial x} = 1}$$

Downstream Gradient **Local Gradient** **Upstream Gradient**

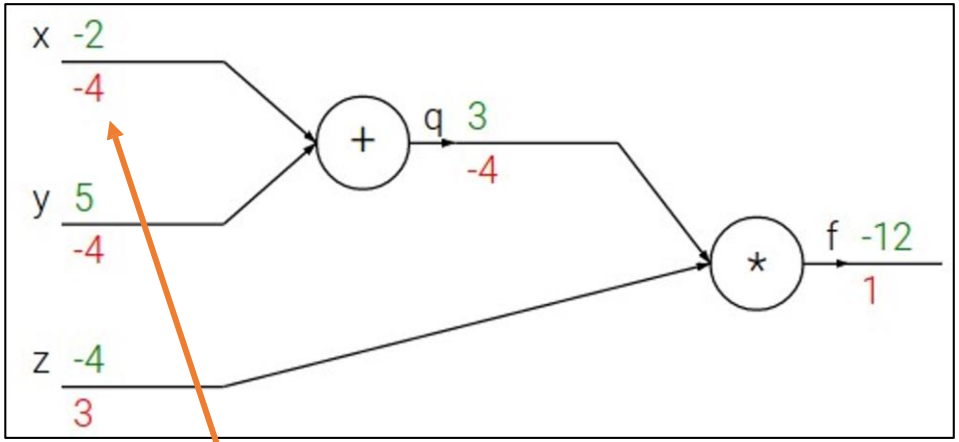
2 . Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$\boxed{q = x + y} \qquad f = qz$$

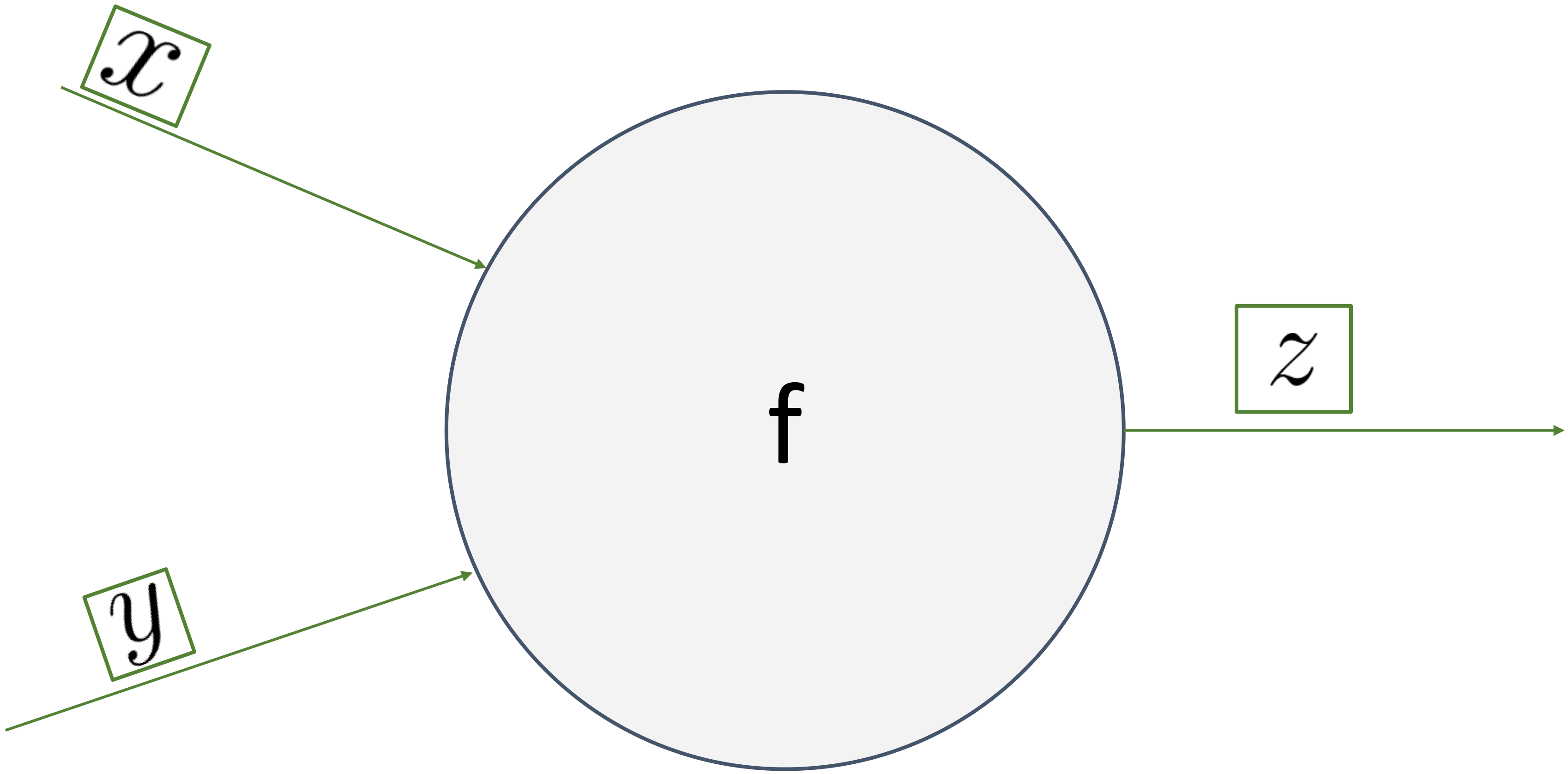
Chain Rule

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial x} = 1}$$

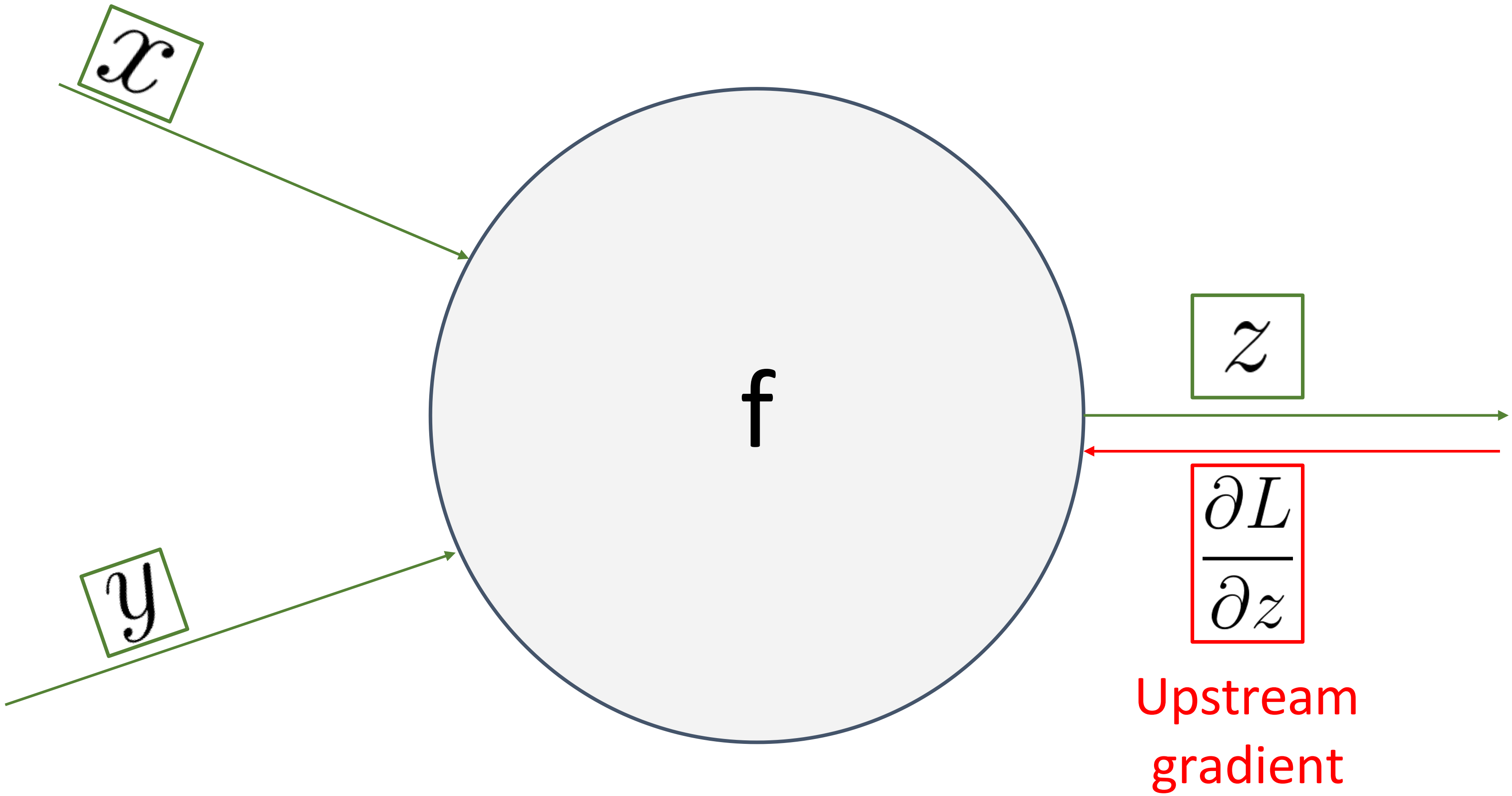
Downstream Gradient **Local Gradient** **Upstream Gradient**

2 . Backward pass: Compute derivatives

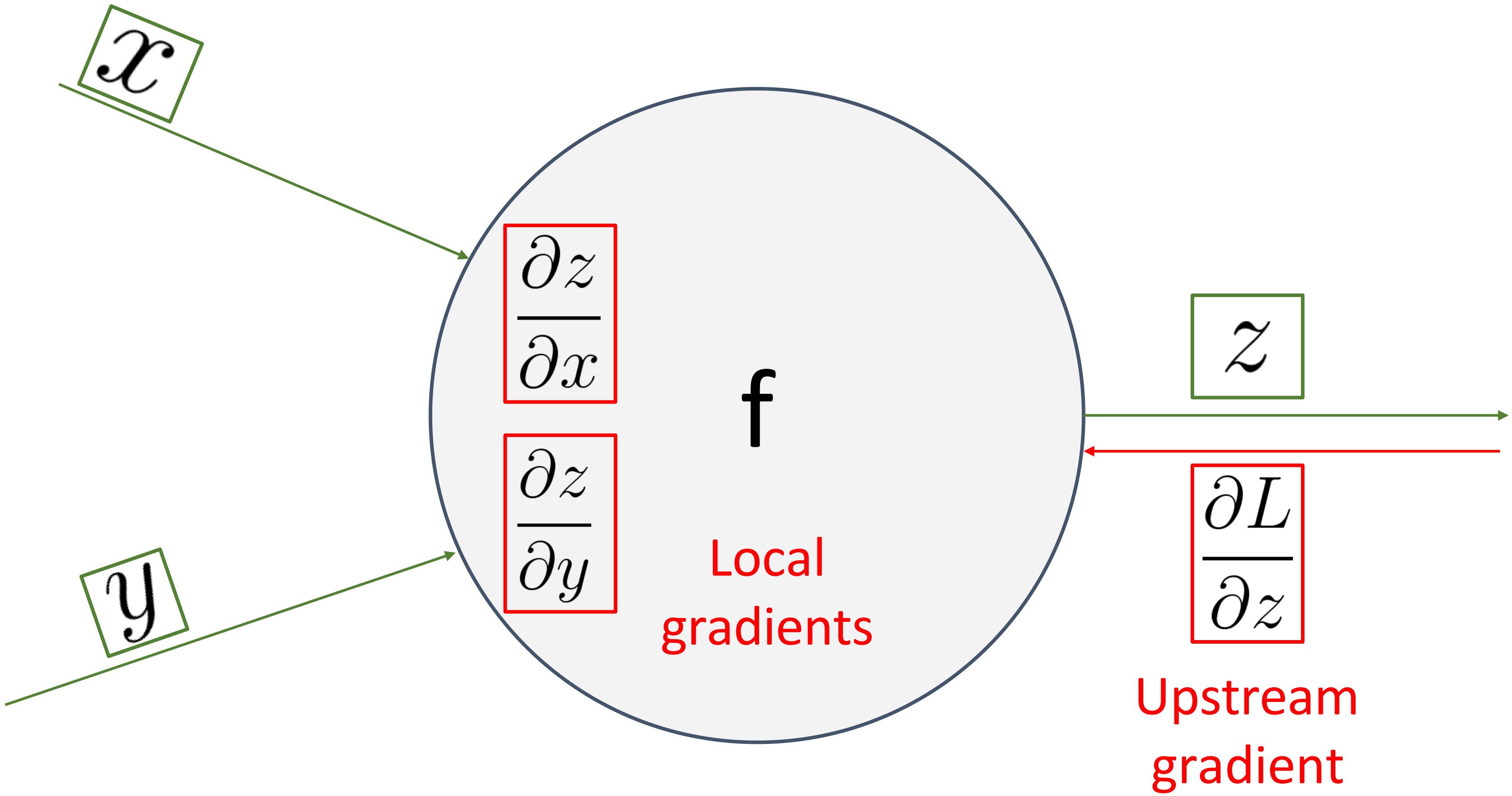
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

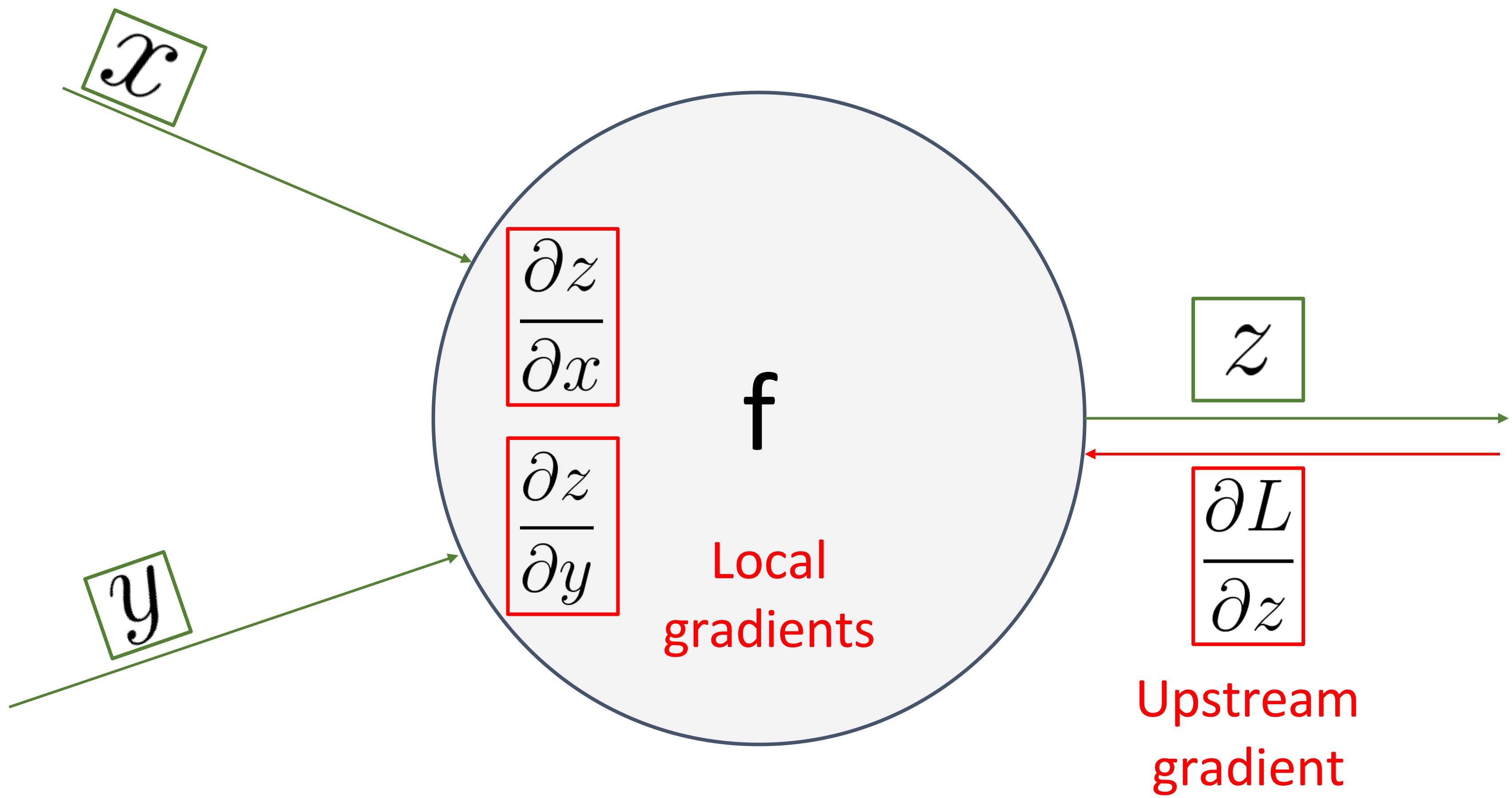


x
y
f
z

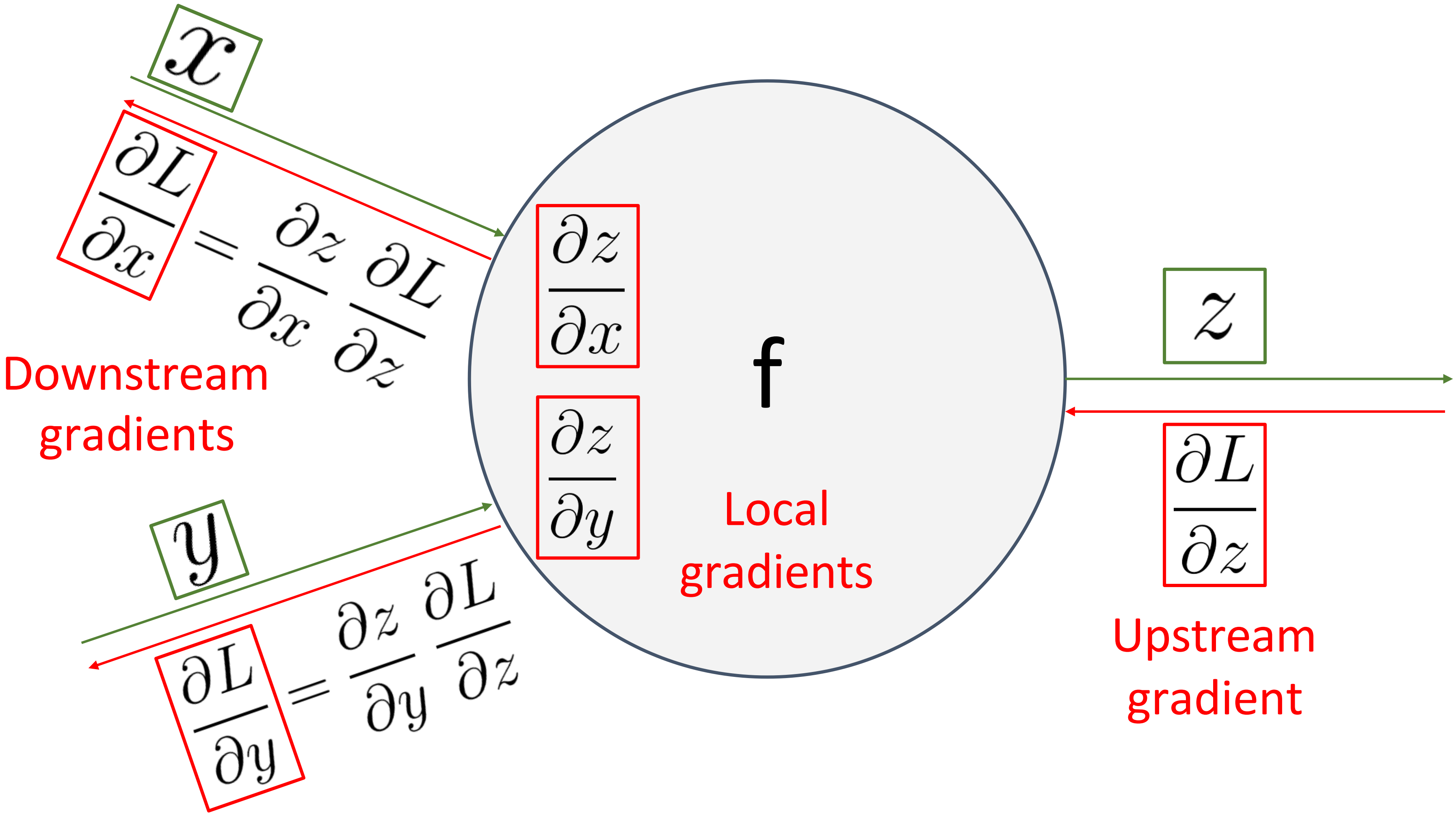


x
y
f
z
$\frac{\partial L}{\partial z}$
Upstream gradient

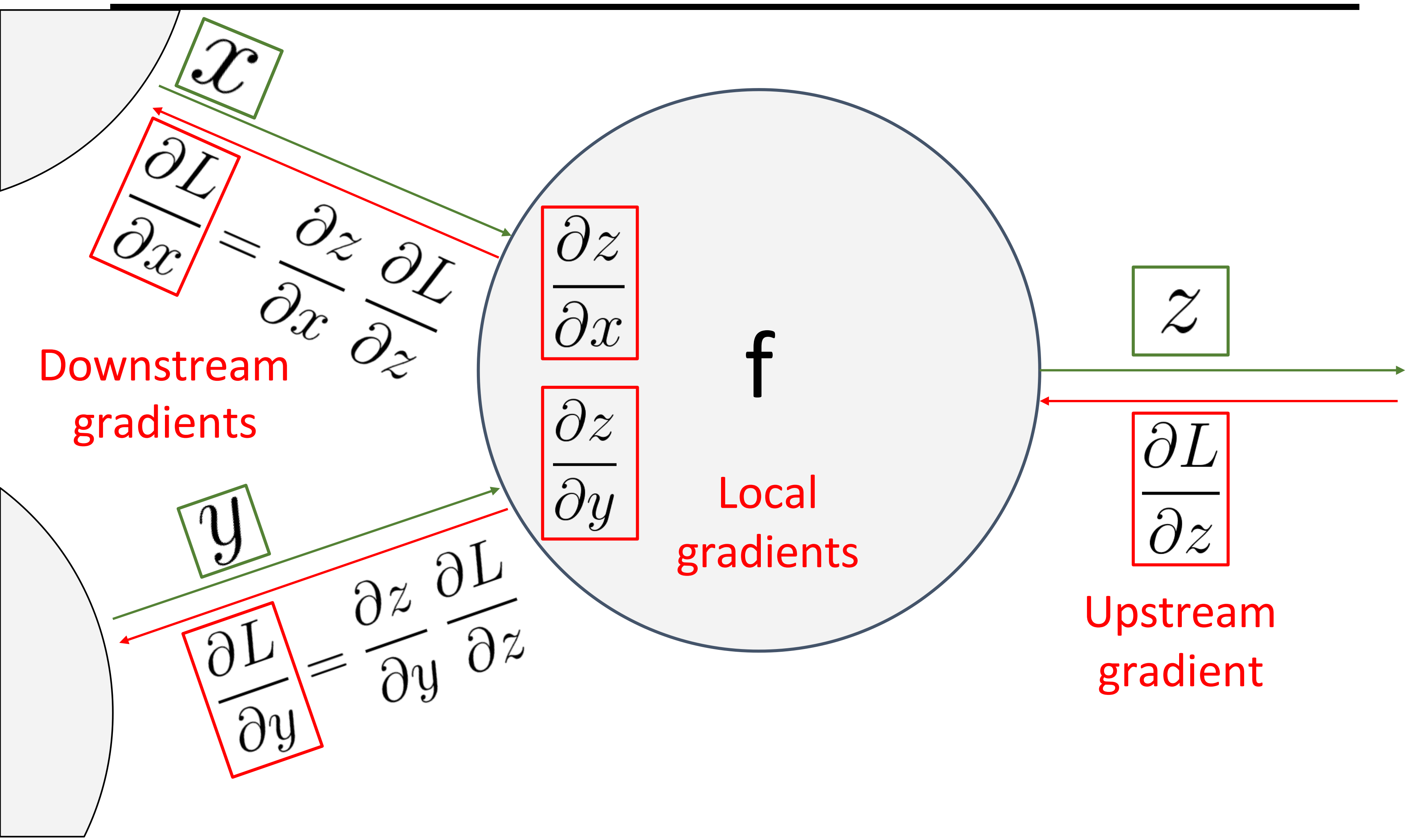

x
y
z
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
Local gradients
$\frac{\partial L}{\partial z}$
Upstream gradient



x
y
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
Local gradients
z
$\frac{\partial L}{\partial z}$
Upstream gradient



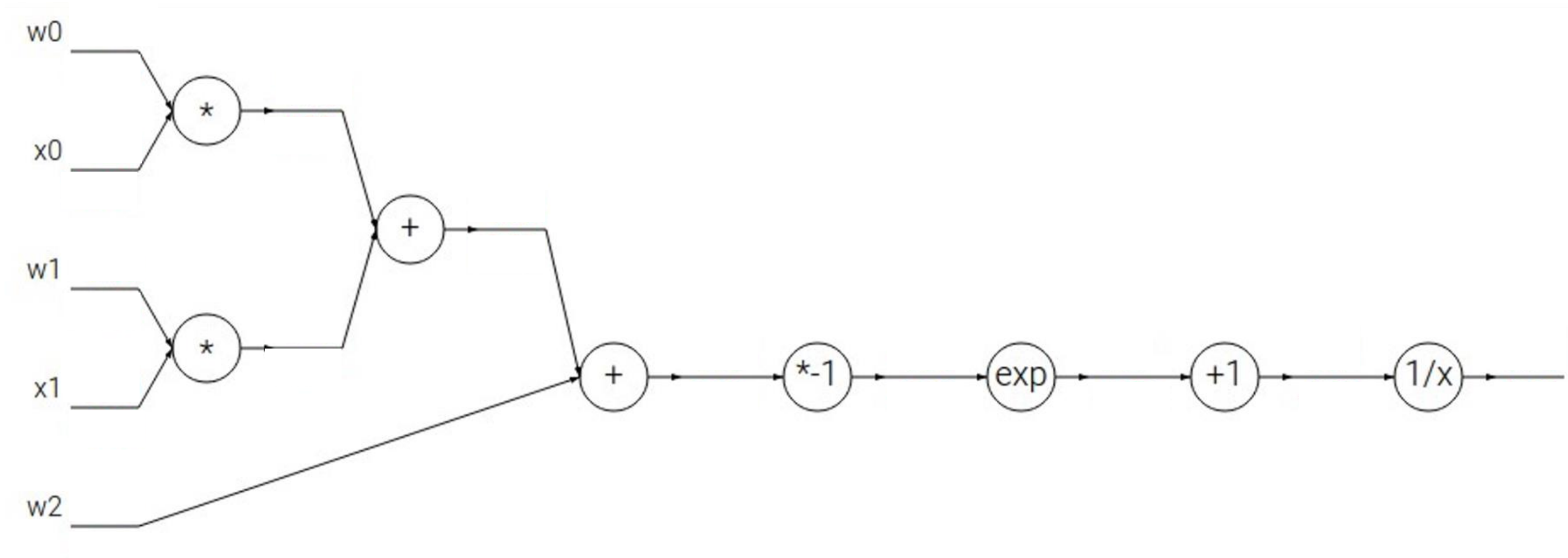
x
$\frac{\partial L}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial L}{\partial z}$
Downstream gradients
y
$\frac{\partial L}{\partial y} = \frac{\partial z}{\partial y}\frac{\partial L}{\partial z}$
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
Local gradients
z
$\frac{\partial L}{\partial z}$
Upstream gradient



x
$\frac{\partial L}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial L}{\partial z}$
Downstream gradients
y
$\frac{\partial L}{\partial y} = \frac{\partial z}{\partial y}\frac{\partial L}{\partial z}$
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
Local gradients
z
$\frac{\partial L}{\partial z}$
Upstream gradient

Another Example

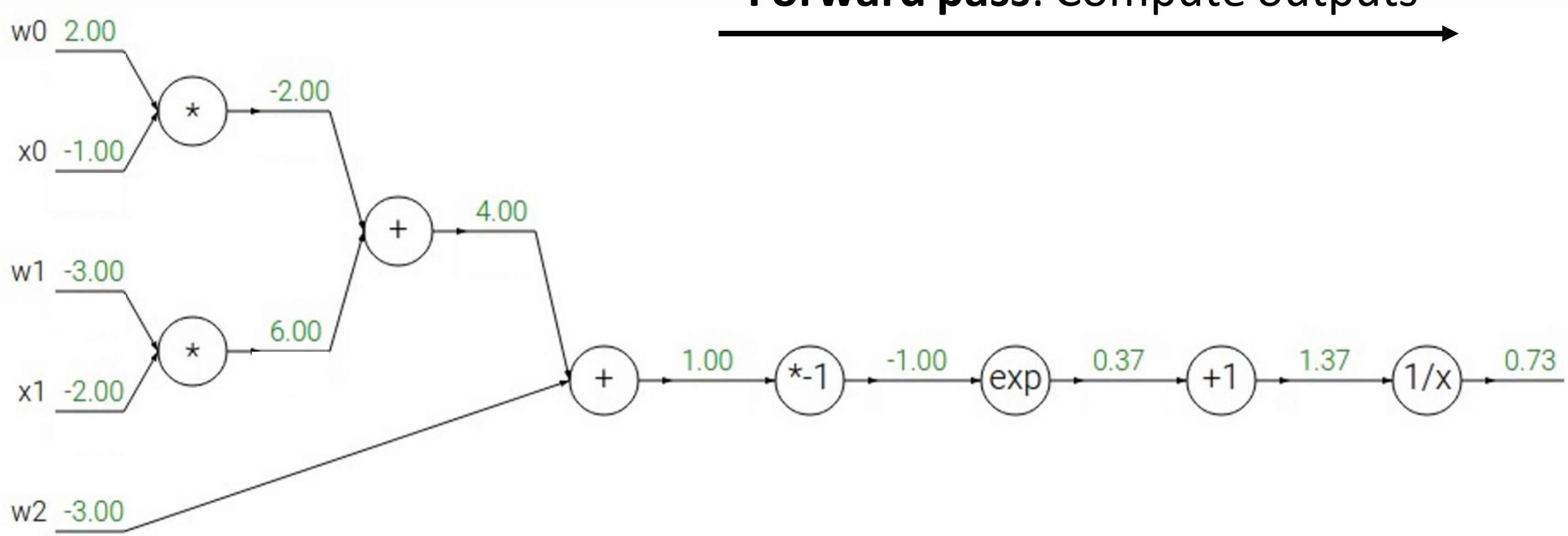
$$f(x,w) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



Another Example

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

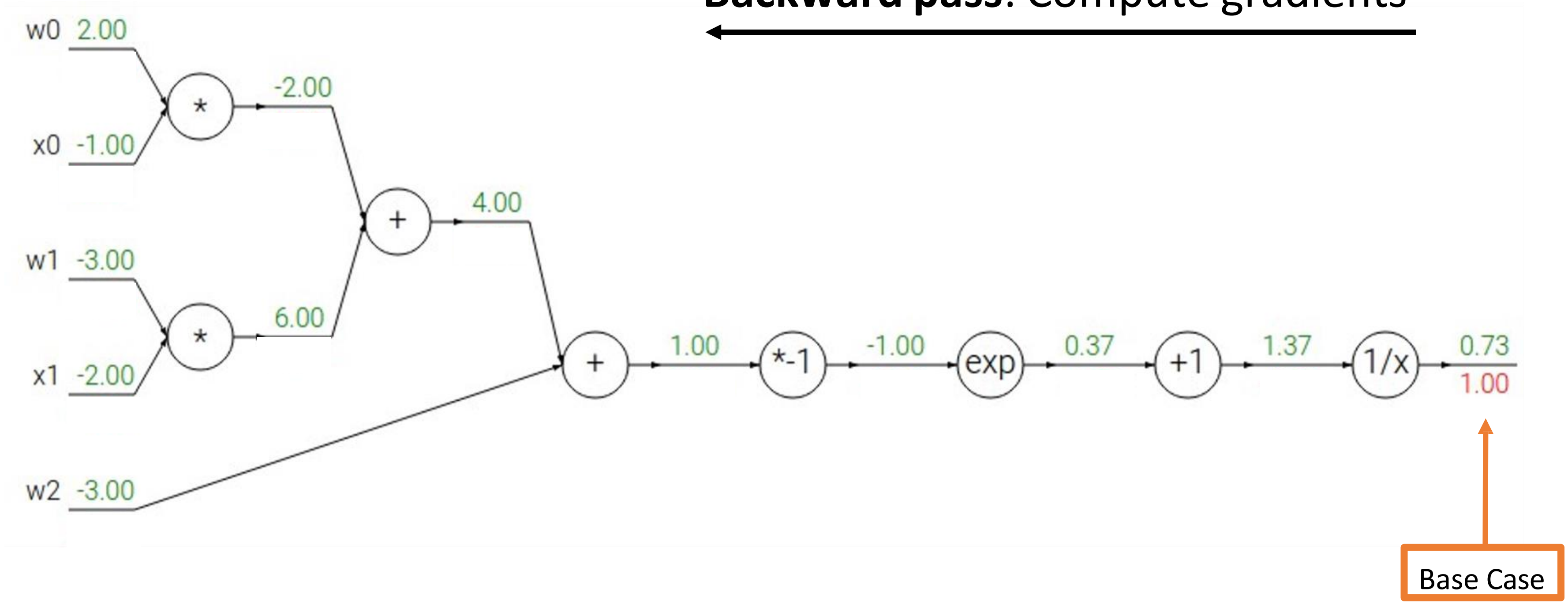
Forward pass: Compute outputs



Another Example

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

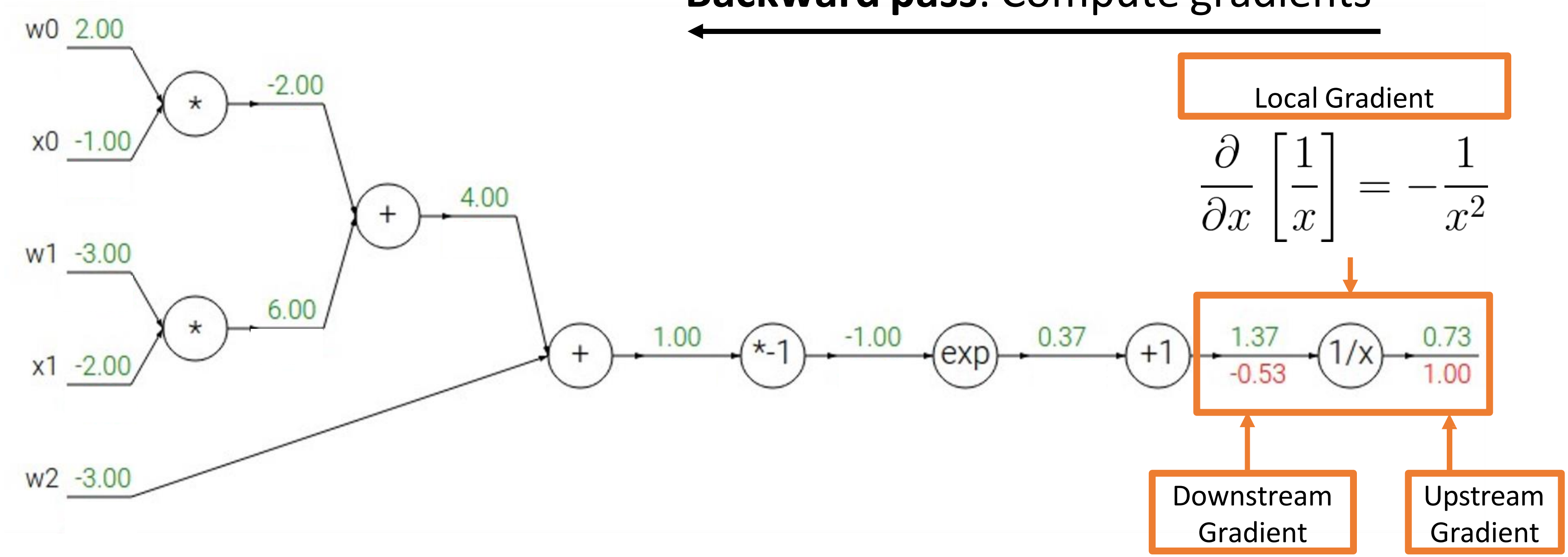
Backward pass: Compute gradients



Another Example

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

Backward pass: Compute gradients

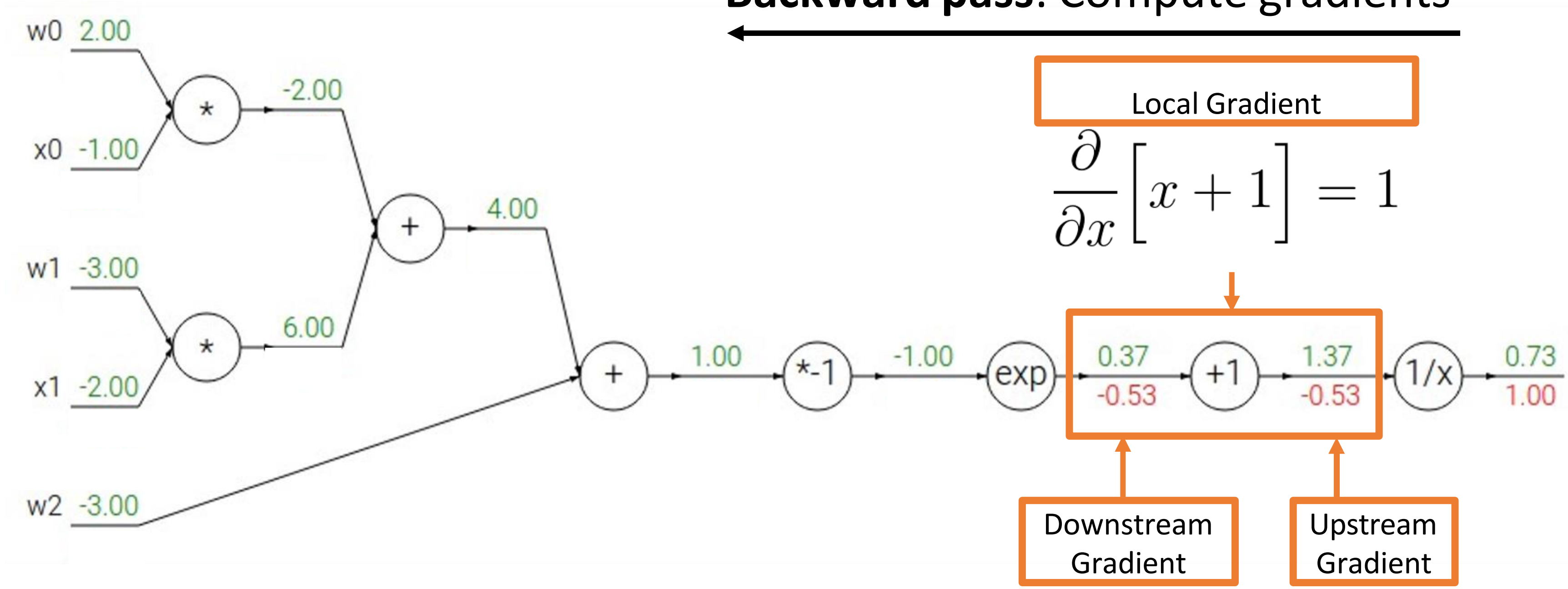


$$\frac{\partial}{\partial x}\left[\frac{1}{x}\right] = -\frac{1}{x^2}$$

Another Example

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

Backward pass: Compute gradients



$$\frac{\partial}{\partial x}\left[x+1\right] = 1$$

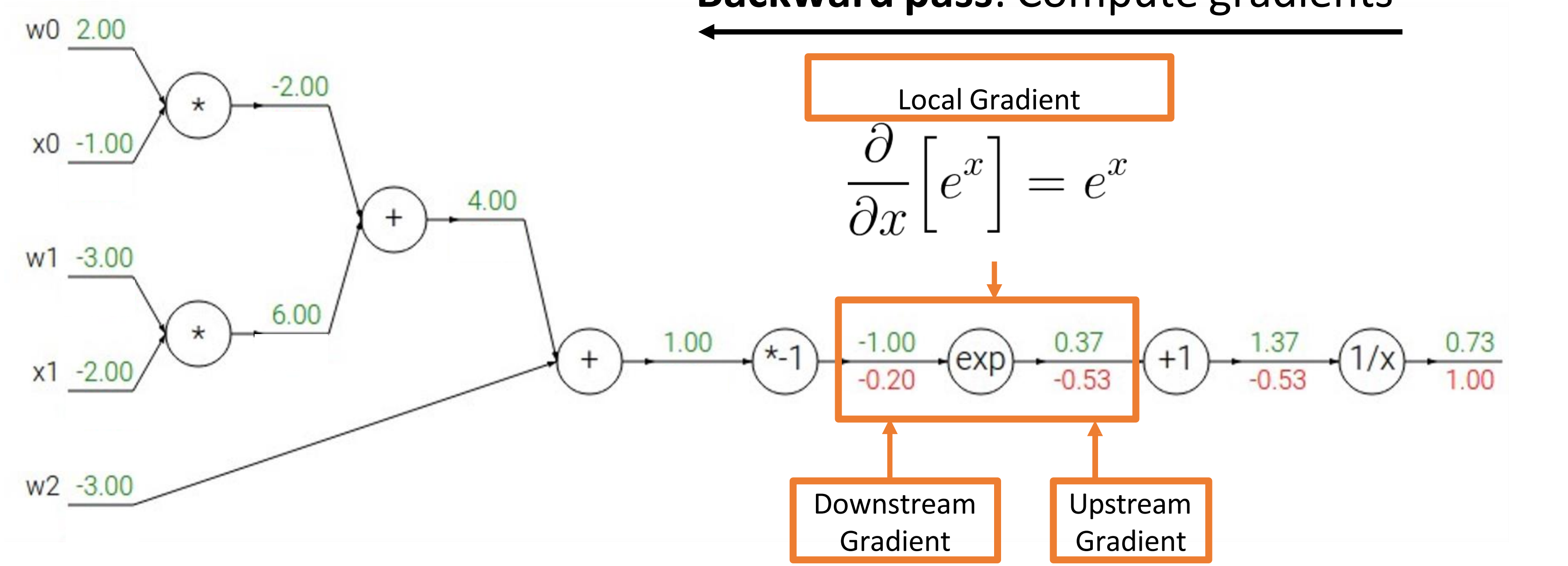
Another Example

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

Backward pass: Compute gradients

Local Gradient

$$\frac{\partial}{\partial x}\left[e^x\right] = e^x$$

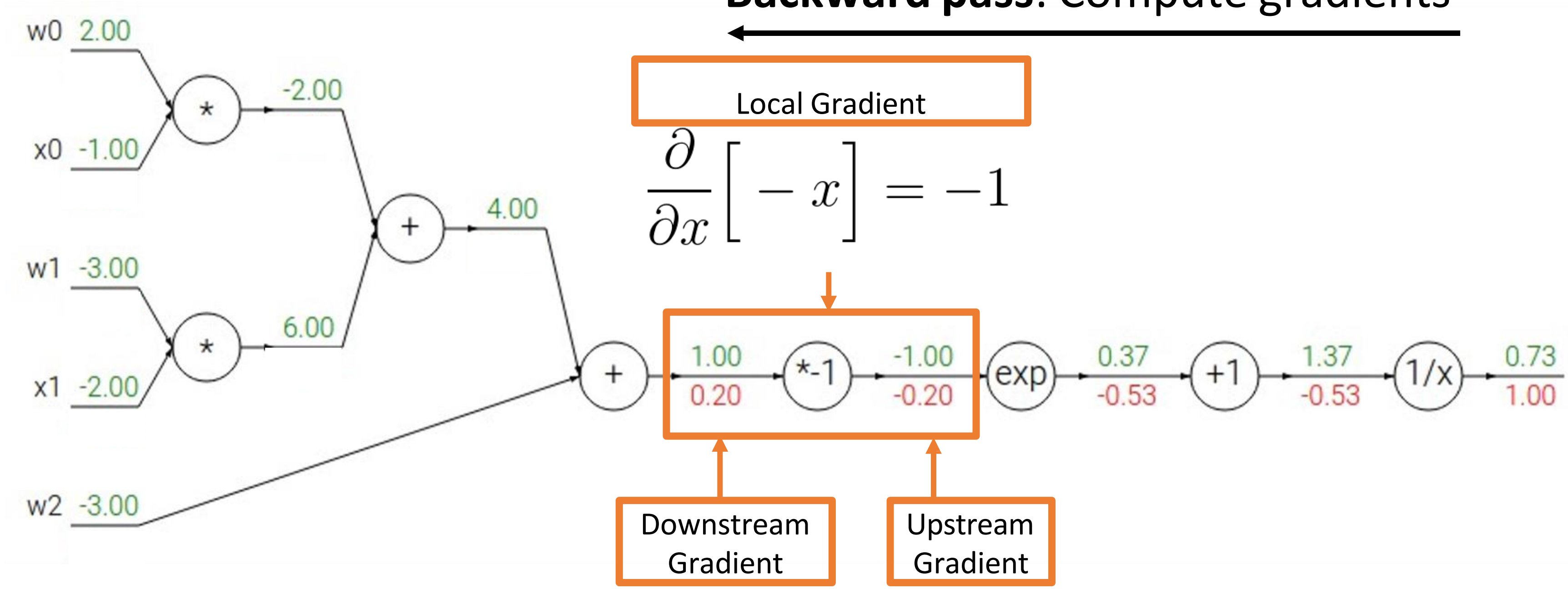


Another Example

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

Backward pass: Compute gradients

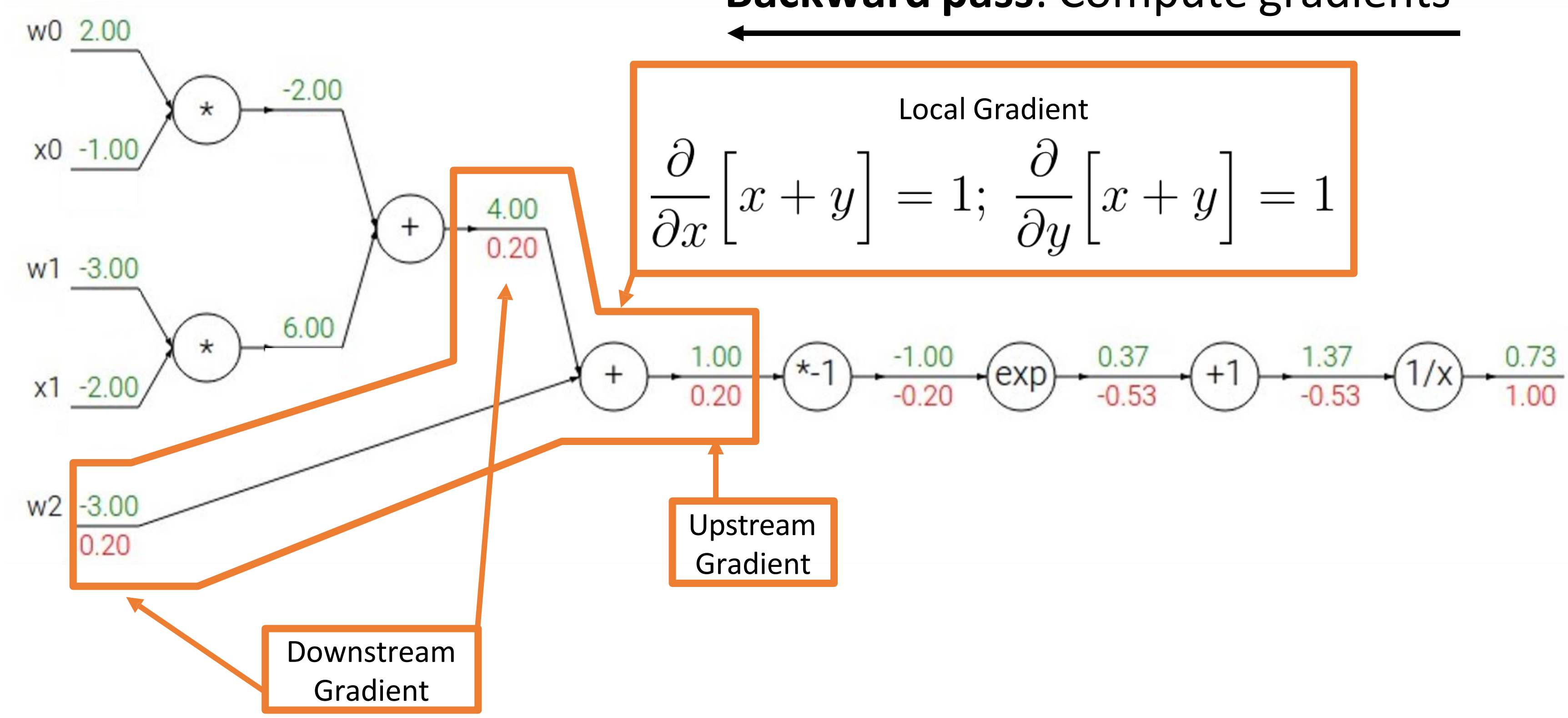
$$\frac{\partial}{\partial x}\left[-x\right] = -1$$



Another Example

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

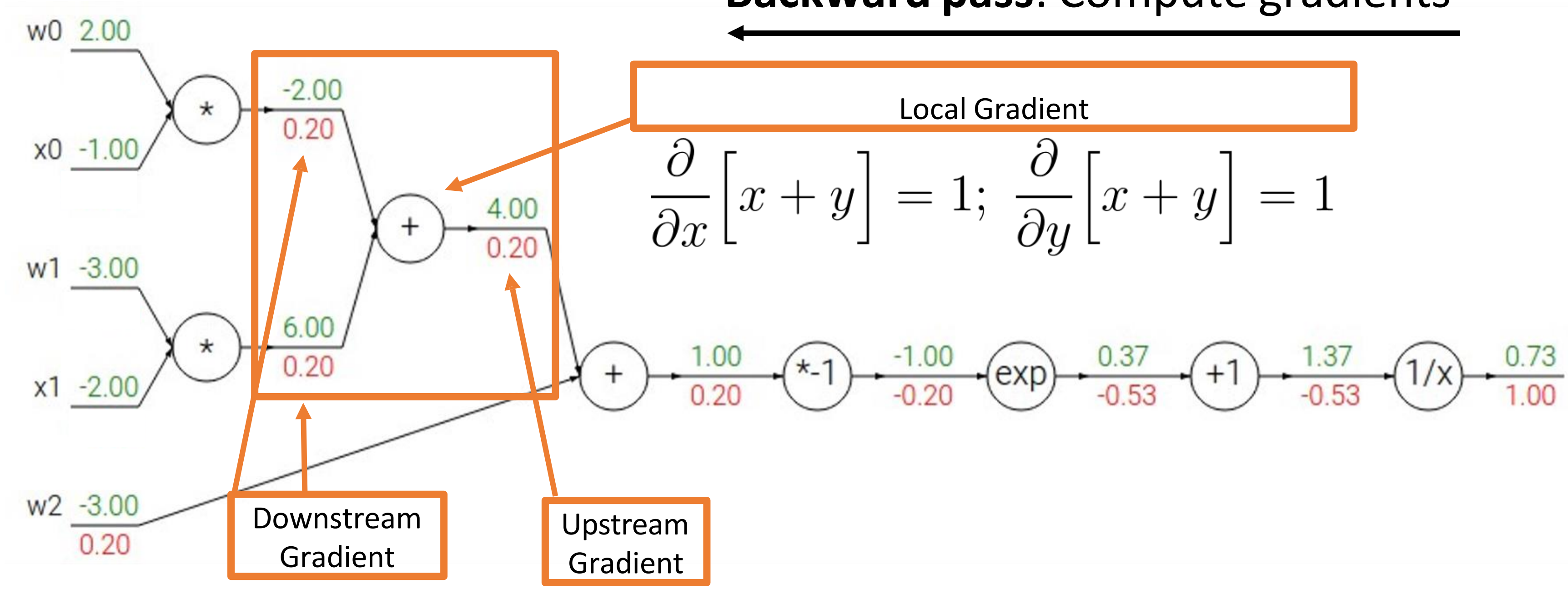
Backward pass: Compute gradients



Another Example

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

Backward pass: Compute gradients

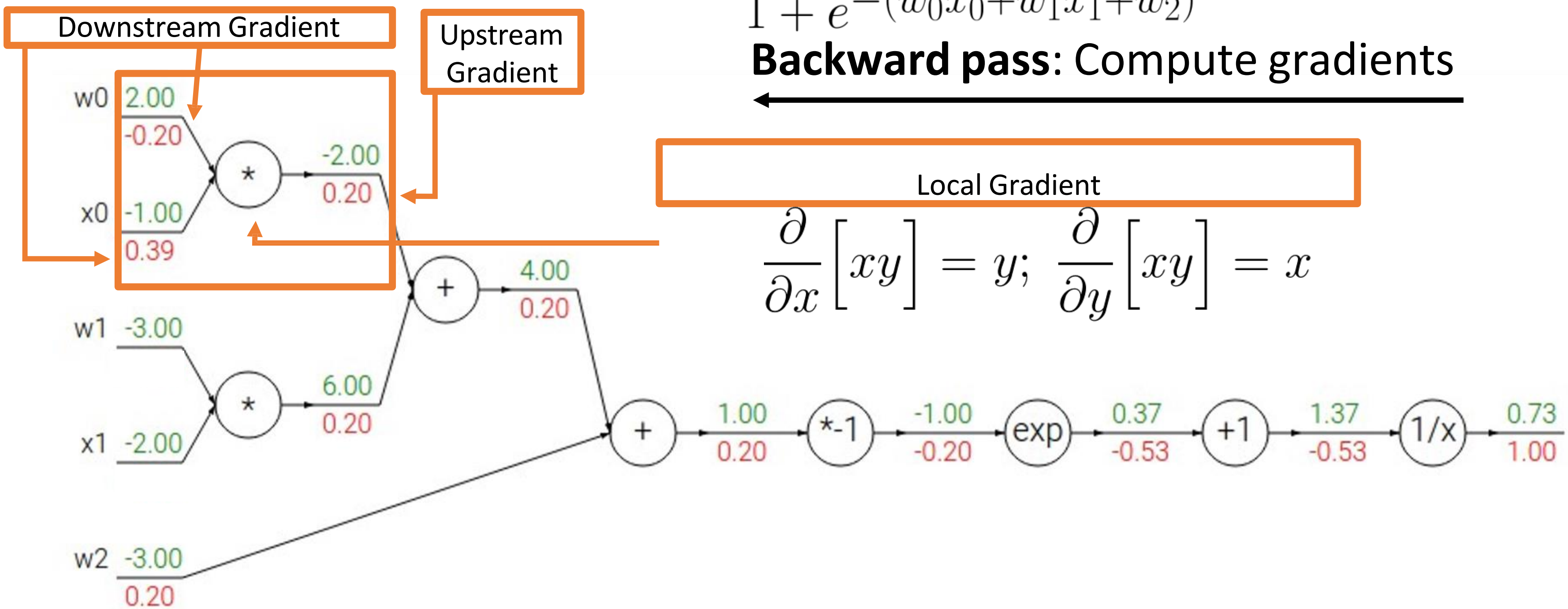


$$\frac{\partial}{\partial x}\left[x+y\right] = 1; \ \frac{\partial}{\partial y}\left[x+y\right] = 1$$

Another Example

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

Backward pass: Compute gradients

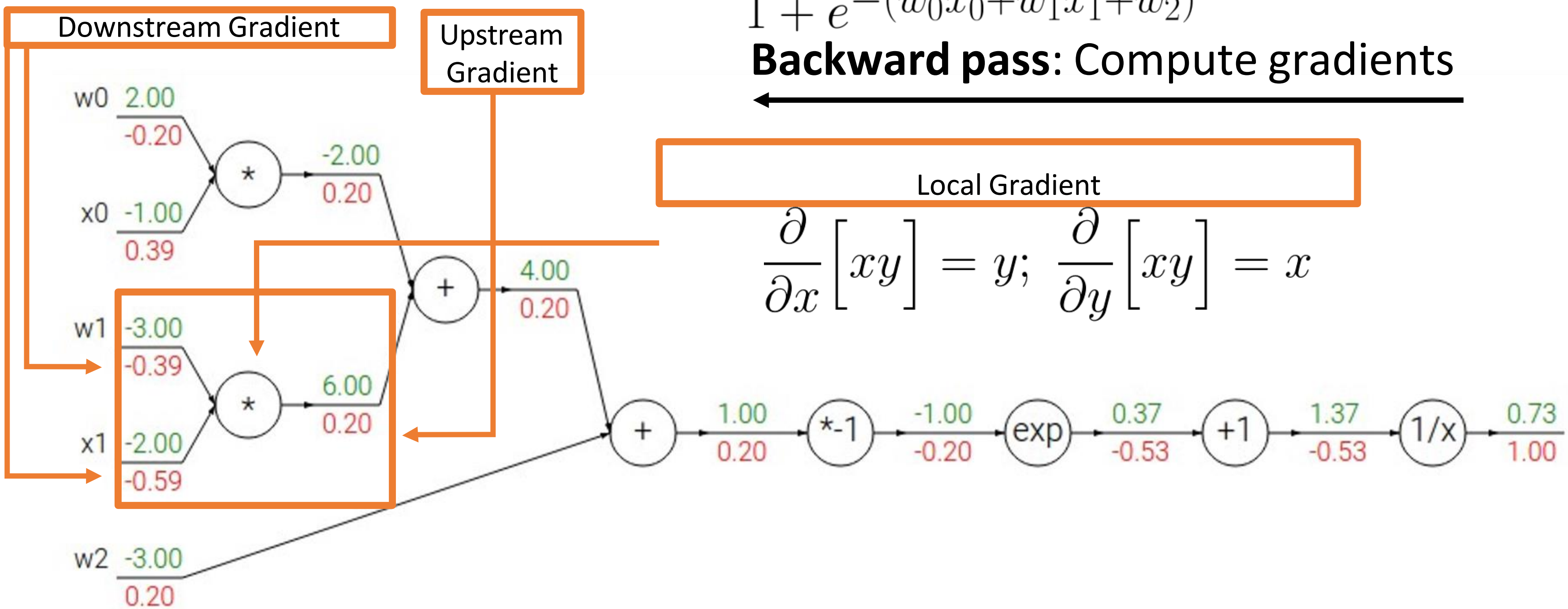


$$\frac{\partial}{\partial x}\left[xy\right] = y; \quad \frac{\partial}{\partial y}\left[xy\right] = x$$

Another Example

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

Backward pass: Compute gradients



$$\frac{\partial}{\partial x}\left[xy\right] = y; \ \frac{\partial}{\partial y}\left[xy\right] = x$$

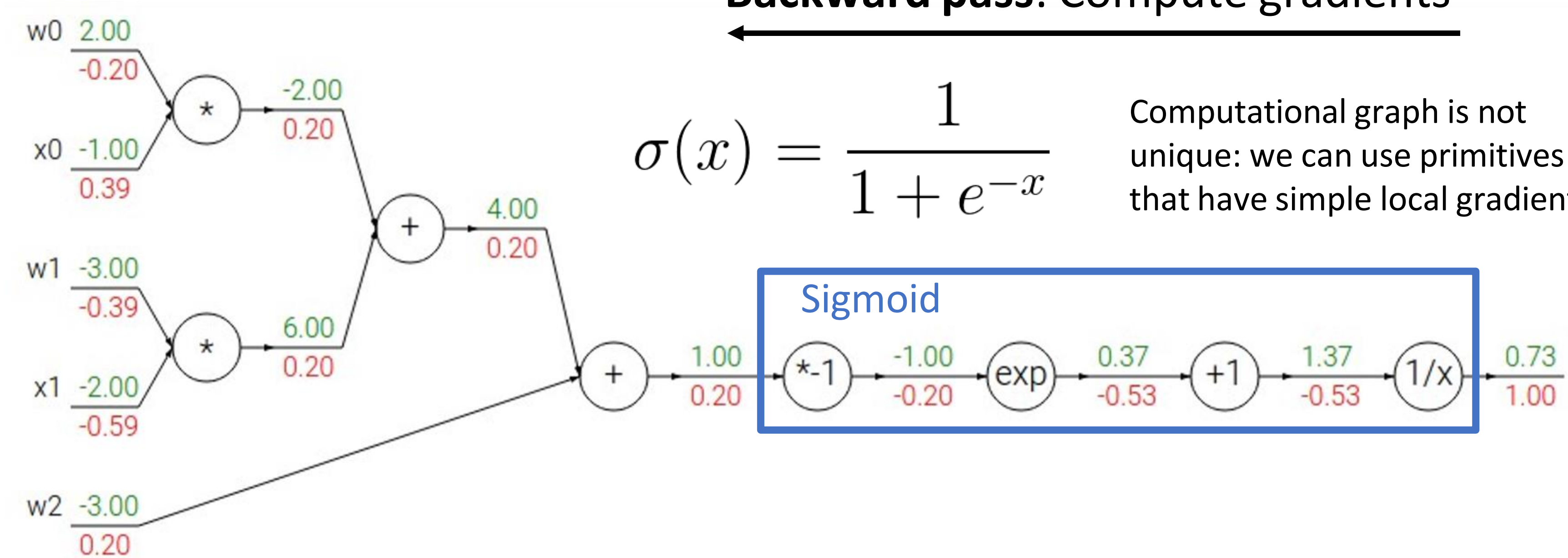
Another Example

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

Backward pass: Compute gradients

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

Computational graph is not unique: we can use primitives that have simple local gradients



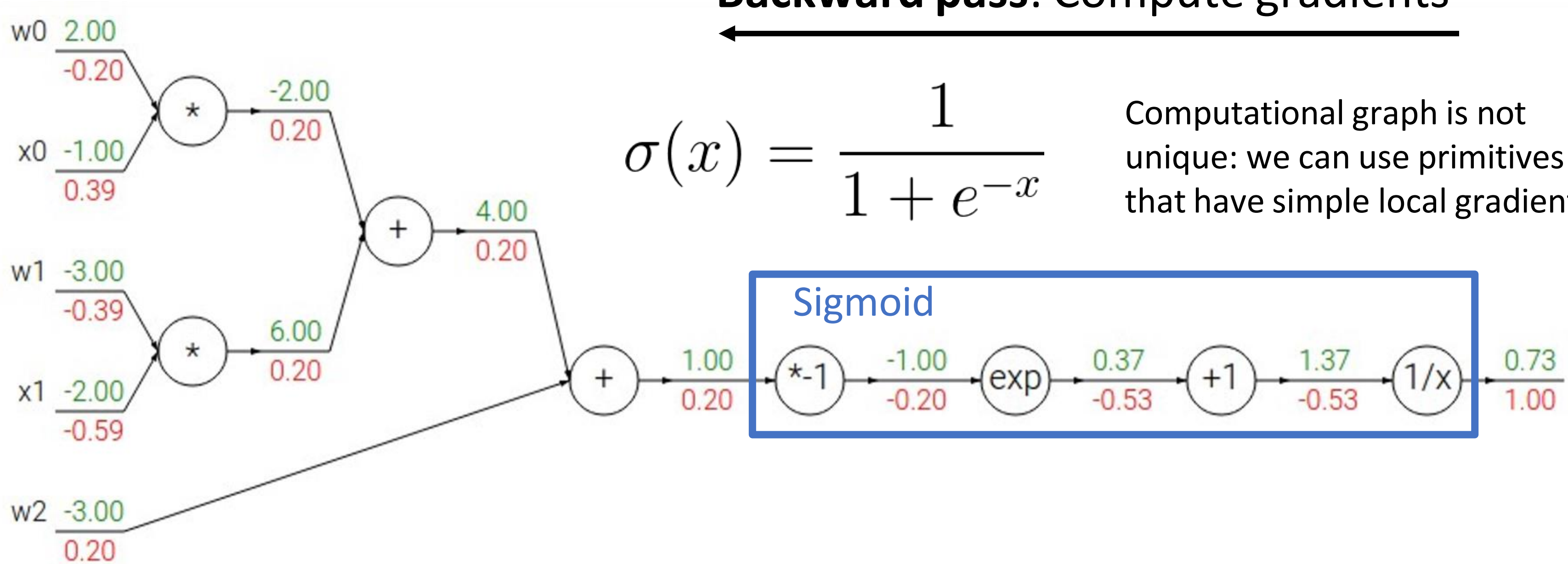
Another Example

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

Backward pass: Compute gradients

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

Computational graph is not unique: we can use primitives that have simple local gradients



Sigmoid local gradient:

$$\frac{\partial}{\partial x}\left[\sigma(x)\right] = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$

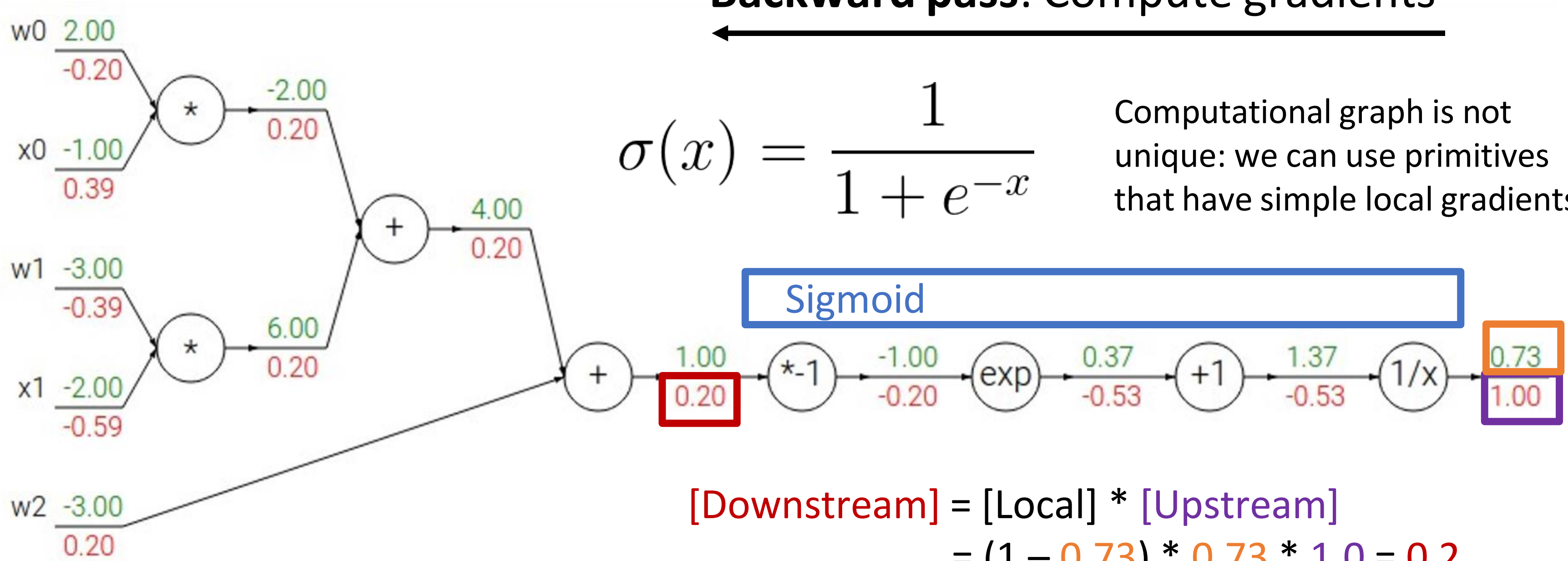
Another Example

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

Backward pass: Compute gradients

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

Computational graph is not unique: we can use primitives that have simple local gradients



[Downstream] = [Local] * [Upstream]

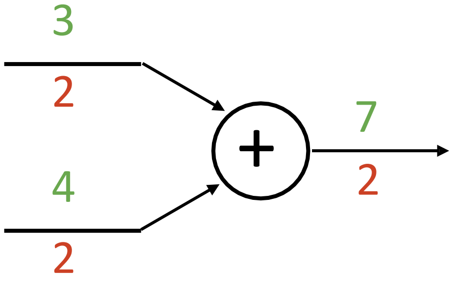
= (1 – 0.73) * 0.73 * 1.0 = 0.2

Sigmoid local gradient:

$$\frac{\partial}{\partial x}\left[\sigma(x)\right] = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$

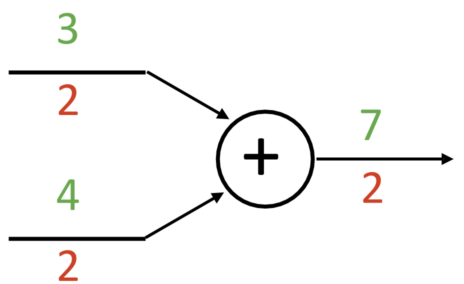
Patterns in Gradient Flow

add gate: gradient distributor

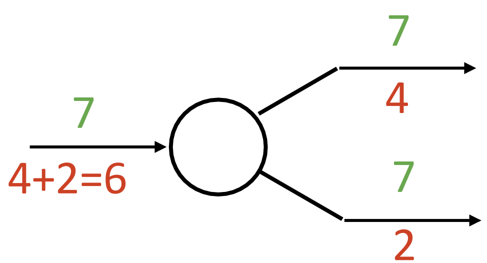


Patterns in Gradient Flow

add gate: gradient distributor

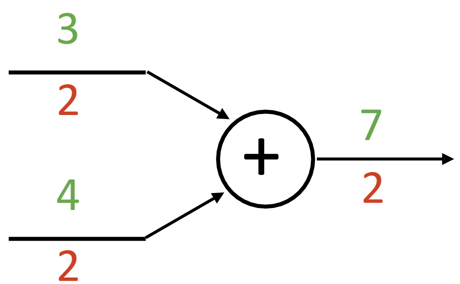


copy gate: gradient adder

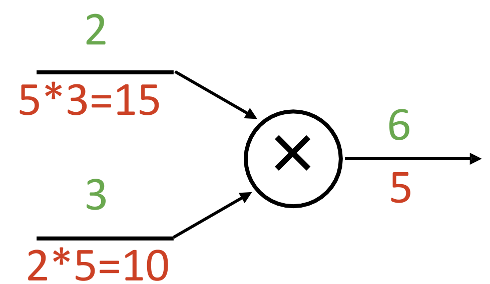


Patterns in Gradient Flow

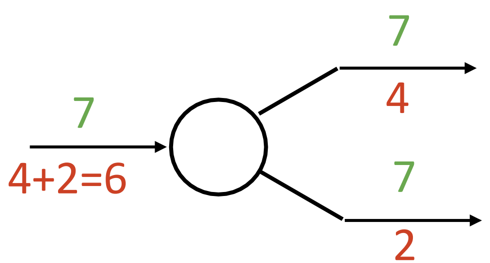
add gate: gradient distributor



mul gate: “swap multiplier”

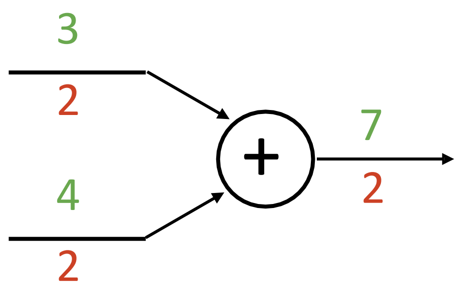


copy gate: gradient adder

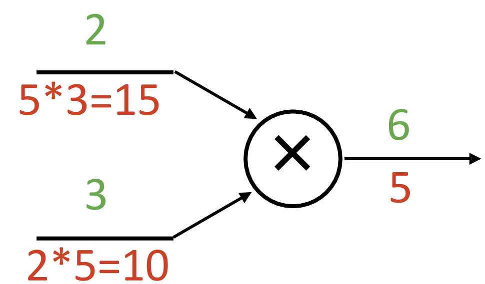


Patterns in Gradient Flow

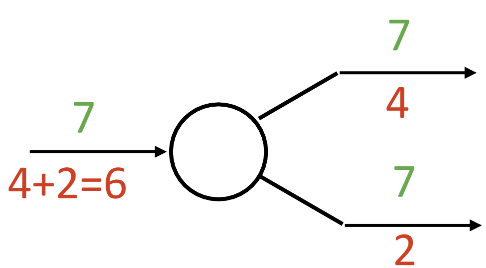
add gate: gradient distributor



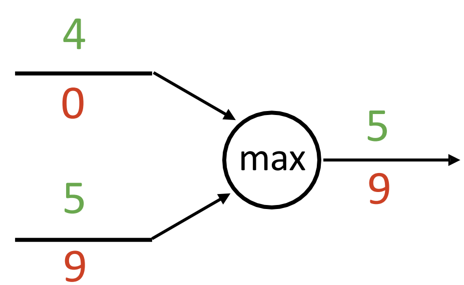
mul gate: “swap multiplier”



copy gate: gradient adder

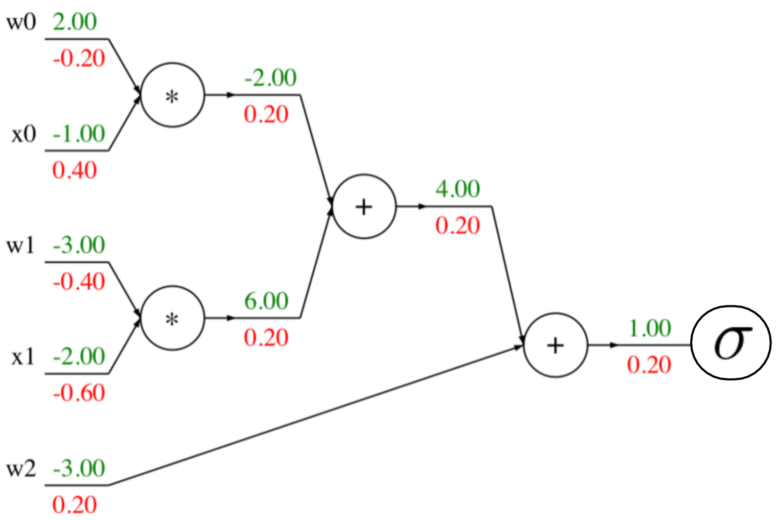


max gate: gradient router



Backprop Implementation:

"Flat" gradient code:

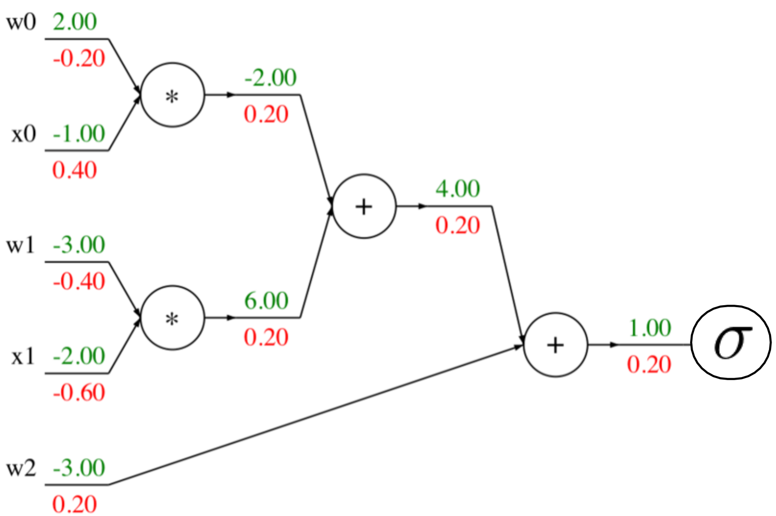


Forward pass:
Compute output

```python
def f(w0, x0, w1, x1, w2):
  s0 = w0 * x0
  s1 = w1 * x1
  s2 = s0 + s1
  s3 = s2 + w2
  L = sigmoid(s3)
```

Backprop Implementation:

”Flat” gradient code:




```
def f(w0, x0, w1, x1, w2):
```

Forward pass:
Compute output

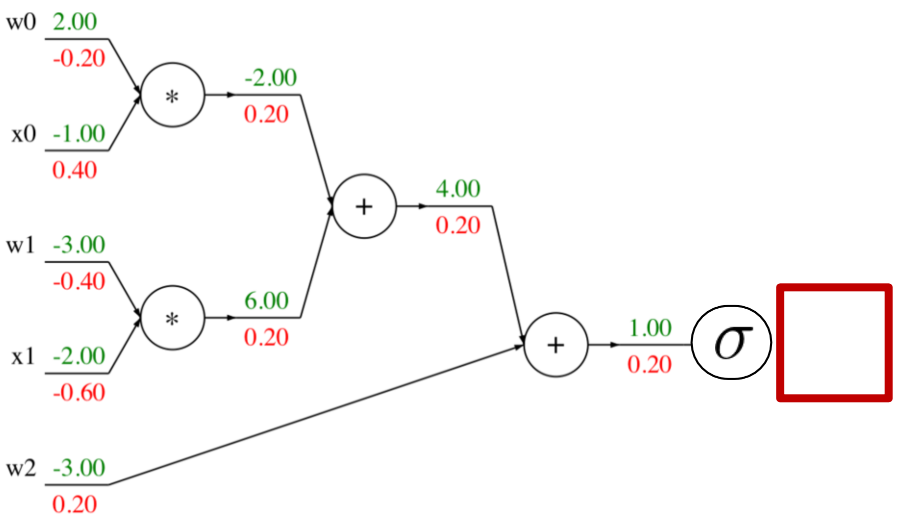
```
s0 = w0 * x0
s1 = w1 * x1
s2 = s0 + s1
s3 = s2 + w2
L = sigmoid(s3)
```

Backward pass:
Compute grads

```
grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0
```

Backprop Implementation:

"Flat" gradient code:



Forward pass: Compute output

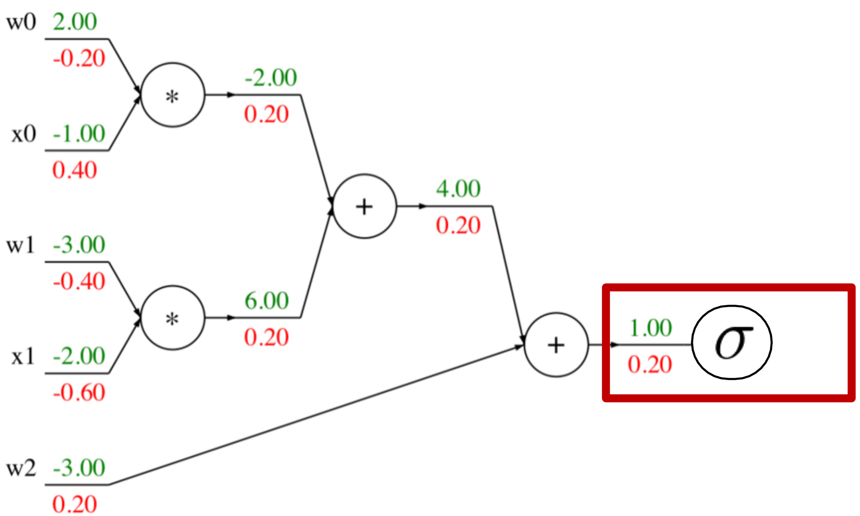
Base case

```
def f(w0, x0, w1, x1, w2):
  s0 = w0 * x0
  s1 = w1 * x1
  s2 = s0 + s1
  s3 = s2 + w2
  L = sigmoid(s3)

  grad_L = 1.0
  grad_s3 = grad_L * (1 - L) * L
  grad_w2 = grad_s3
  grad_s2 = grad_s3
  grad_s0 = grad_s2
  grad_s1 = grad_s2
  grad_w1 = grad_s1 * x1
  grad_x1 = grad_s1 * w1
  grad_w0 = grad_s0 * x0
  grad_x0 = grad_s0 * w0
```

Backprop Implementation:

"Flat" gradient code:



Forward pass:
Compute output

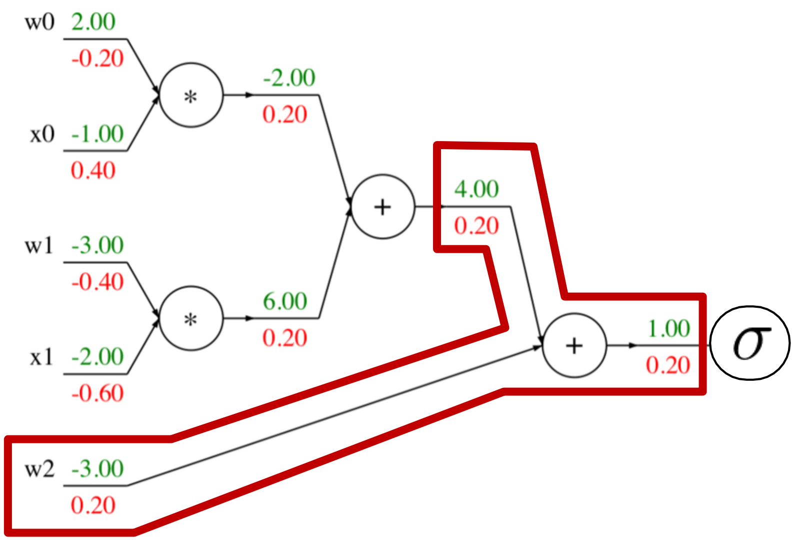
Sigmoid

```
def f(w0, x0, w1, x1, w2):
    s0 = w0 * x0
    s1 = w1 * x1
    s2 = s0 + s1
    s3 = s2 + w2
    L = sigmoid(s3)

    grad_L = 1.0
    grad_s3 = grad_L * (1 - L) * L
    grad_w2 = grad_s3
    grad_s2 = grad_s3
    grad_s0 = grad_s2
    grad_s1 = grad_s2
    grad_w1 = grad_s1 * x1
    grad_x1 = grad_s1 * w1
    grad_w0 = grad_s0 * x0
    grad_x0 = grad_s0 * w0
```

Backprop Implementation:

"Flat" gradient code:



Forward pass:
Compute output

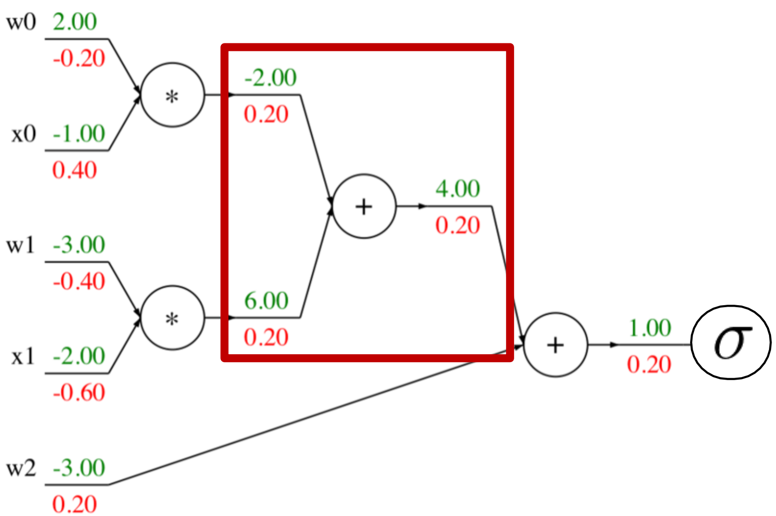
Add

```
def f(w0, x0, w1, x1, w2):
  s0 = w0 * x0
  s1 = w1 * x1
  s2 = s0 + s1
  s3 = s2 + w2
  L = sigmoid(s3)

  grad_L = 1.0
  grad_s3 = grad_L * (1 - L) * L
  grad_w2 = grad_s3
  grad_s2 = grad_s3
  grad_s0 = grad_s2
  grad_s1 = grad_s2
  grad_w1 = grad_s1 * x1
  grad_x1 = grad_s1 * w1
  grad_w0 = grad_s0 * x0
  grad_x0 = grad_s0 * w0
```

Backprop Implementation:

"Flat" gradient code:




```
def f(w0, x0, w1, x1, w2):
```

Forward pass:
Compute output

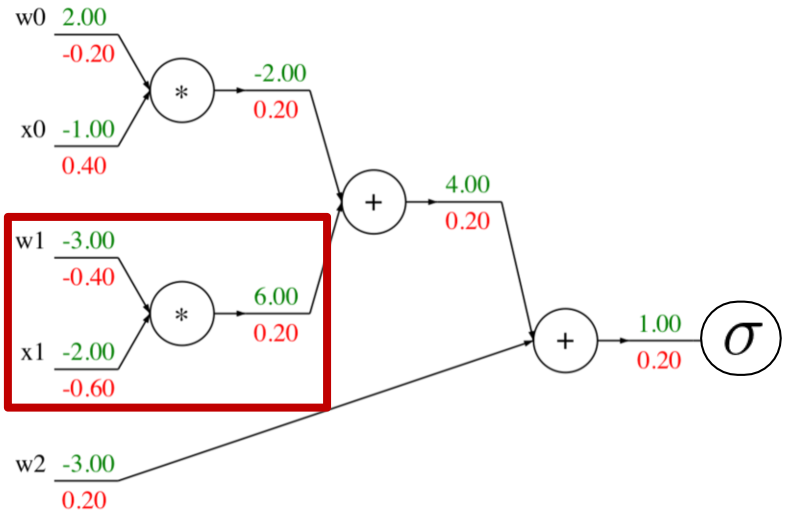
```
    s0 = w0 * x0
    s1 = w1 * x1
    s2 = s0 + s1
    s3 = s2 + w2
    L = sigmoid(s3)
```

Add

```
    grad_L = 1.0
    grad_s3 = grad_L * (1 - L) * L
    grad_w2 = grad_s3
    grad_s2 = grad_s3
    grad_s0 = grad_s2
    grad_s1 = grad_s2
    grad_w1 = grad_s1 * x1
    grad_x1 = grad_s1 * w1
    grad_w0 = grad_s0 * x0
    grad_x0 = grad_s0 * w0
```

Backprop Implementation:

"Flat" gradient code:



Forward pass:
Compute output

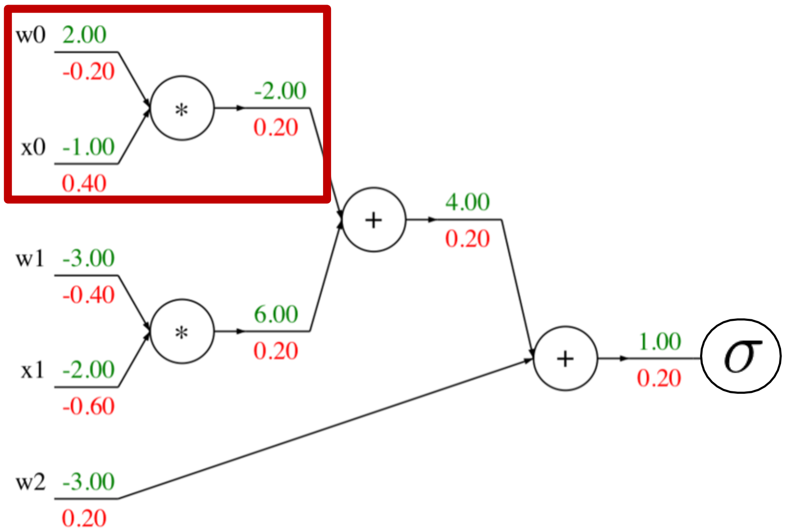
```
def f(w0, x0, w1, x1, w2):
    s0 = w0 * x0
    s1 = w1 * x1
    s2 = s0 + s1
    s3 = s2 + w2
    L = sigmoid(s3)

    grad_L = 1.0
    grad_s3 = grad_L * (1 - L) * L
    grad_w2 = grad_s3
    grad_s2 = grad_s3
    grad_s0 = grad_s2
    grad_s1 = grad_s2
    grad_w1 = grad_s1 * x1
    grad_x1 = grad_s1 * w1
    grad_w0 = grad_s0 * x0
    grad_x0 = grad_s0 * w0
```

Multiply

Backprop Implementation:

”Flat” gradient code:



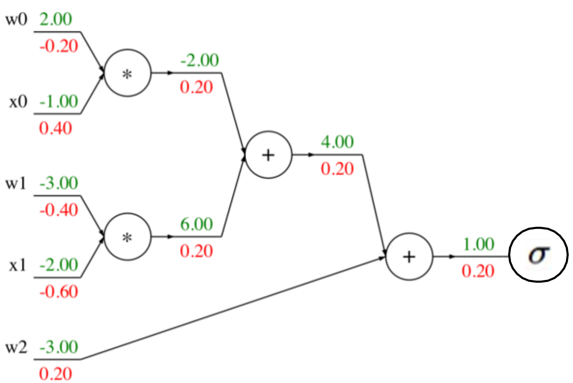
Forward pass:
Compute output

```
def f(w0, x0, w1, x1, w2):
    s0 = w0 * x0
    s1 = w1 * x1
    s2 = s0 + s1
    s3 = s2 + w2
    L = sigmoid(s3)

    grad_L = 1.0
    grad_s3 = grad_L * (1 - L) * L
    grad_w2 = grad_s3
    grad_s2 = grad_s3
    grad_s0 = grad_s2
    grad_s1 = grad_s2
    grad_w1 = grad_s1 * x1
    grad_x1 = grad_s1 * w1
    grad_w0 = grad_s0 * x0
    grad_x0 = grad_s0 * w0
```

Multiply

Backprop Implementation: Modular API



Graph (or Net) object *(rough pseudo code)*

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

So far: backprop with scalars

What about vector-valued functions?

Recap: Vector Derivatives

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If x changes by a small amount, how much will y change?

Recap: Vector Derivatives

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If x changes by a small amount, how much will y change?

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will y change?

Recap: Vector Derivatives

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If x changes by a small amount, how much will y change?

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will y change?

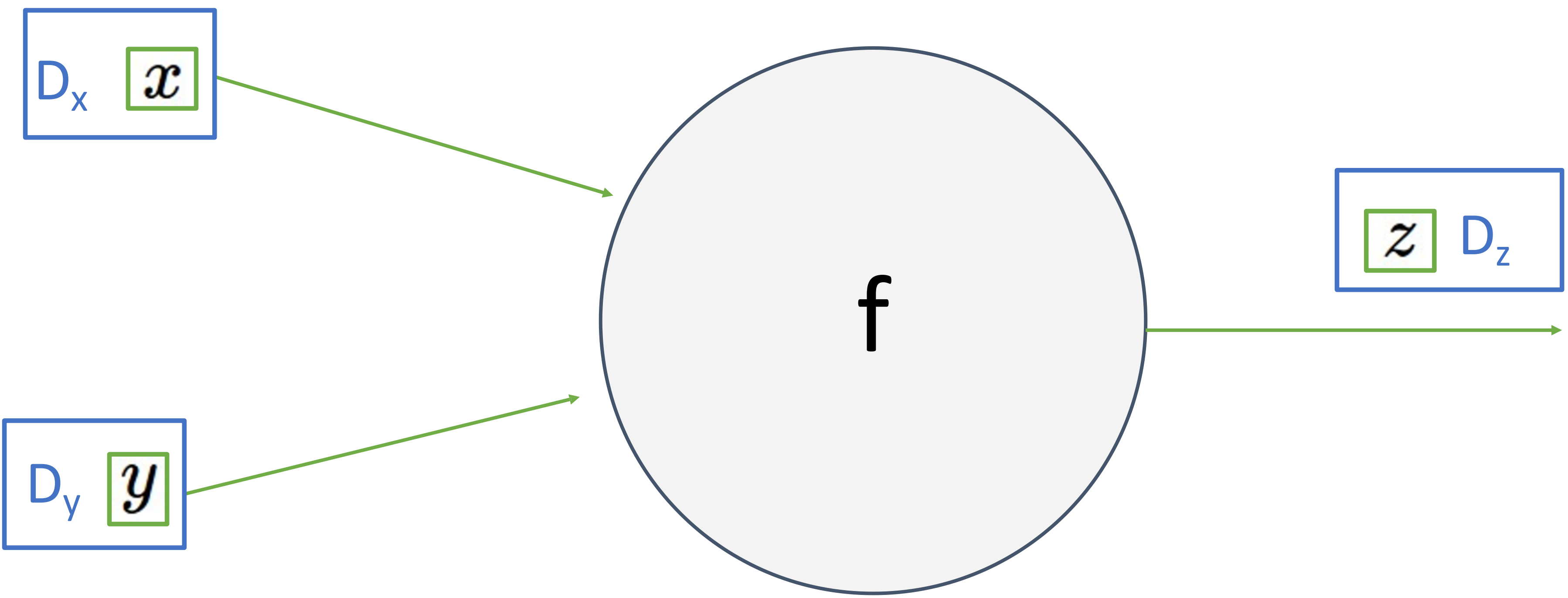
$$x \in \mathbb{R}^N, y \in \mathbb{R}^M$$

Derivative is **Jacobian**:

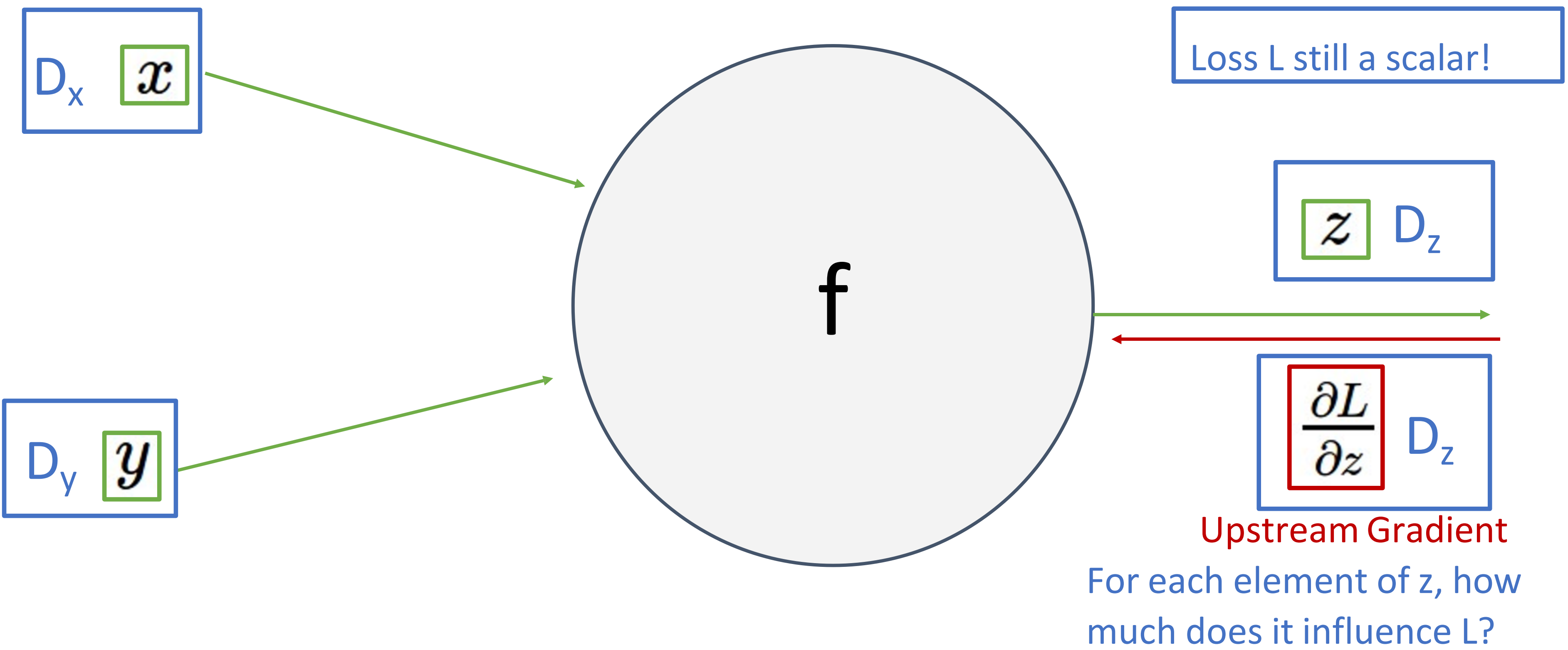
$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will each element of y change?

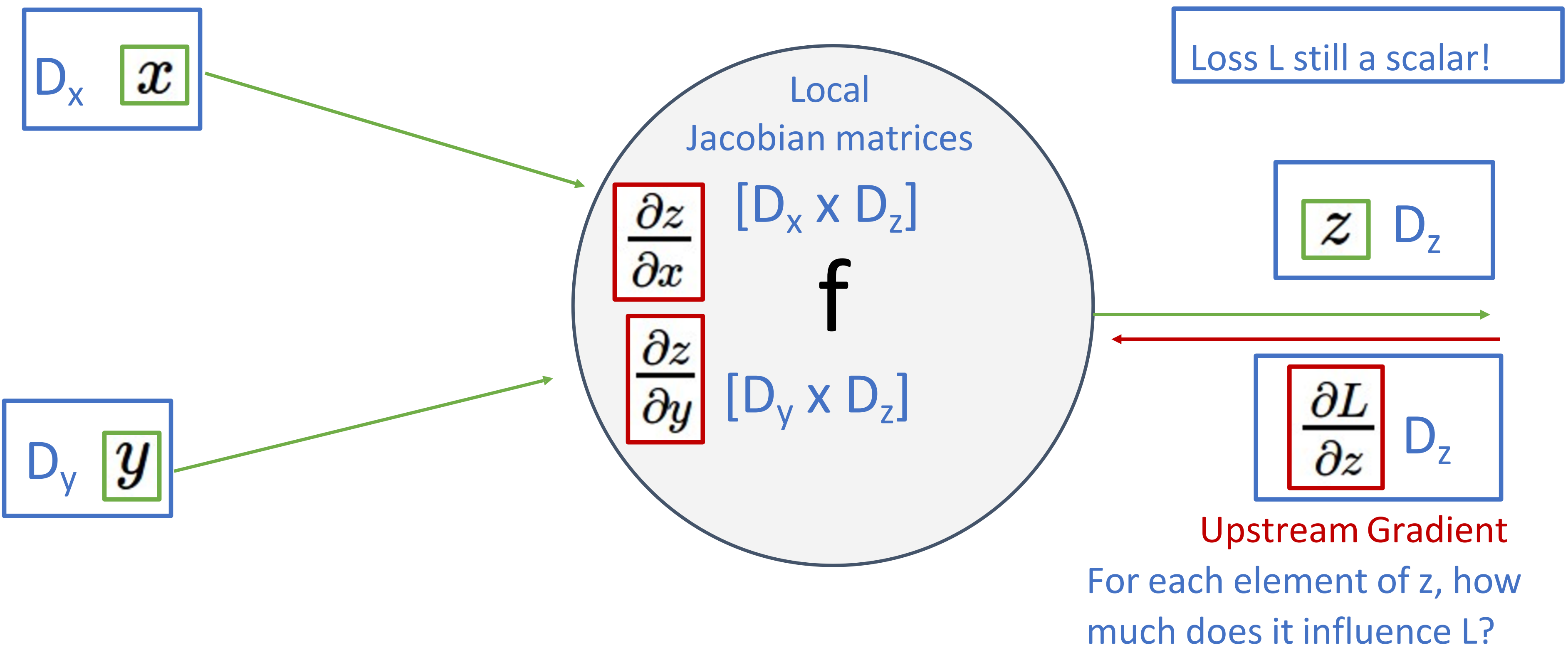
Backprop with Vectors



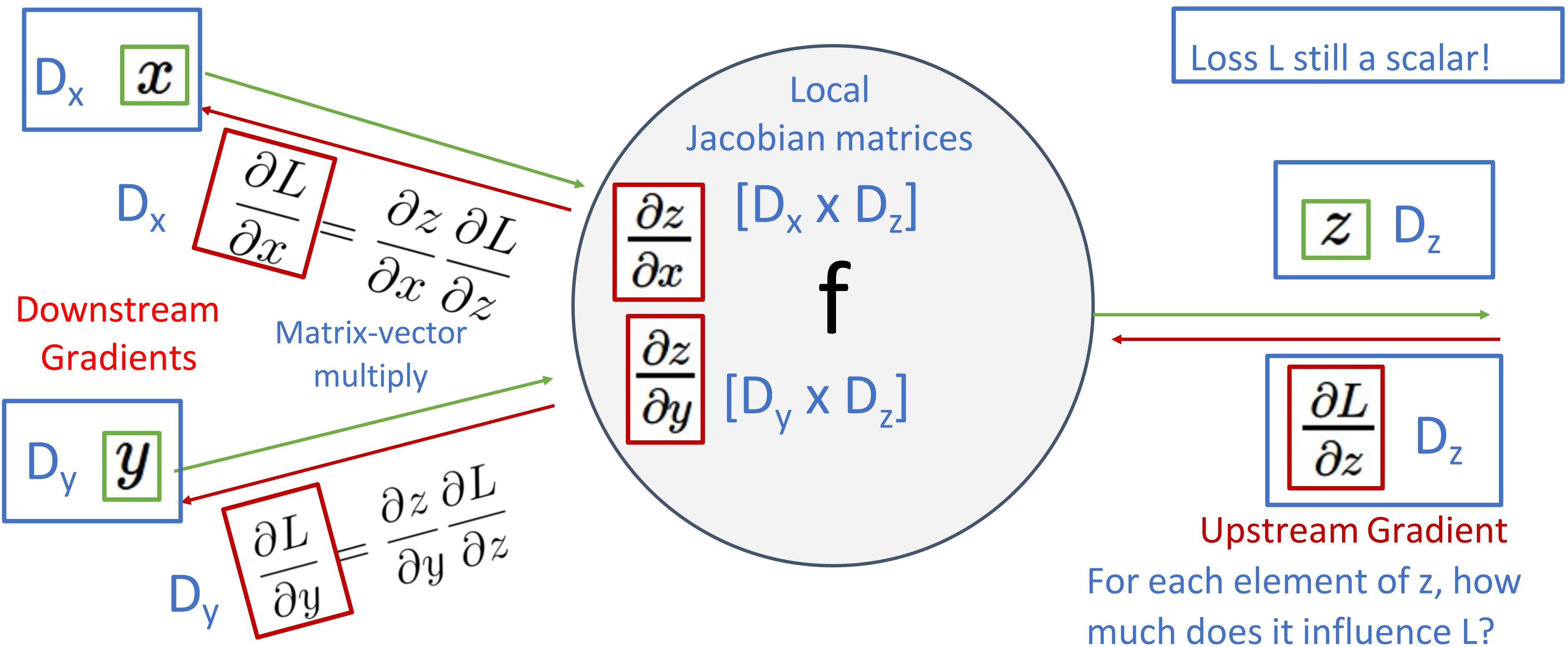
Backprop with Vectors



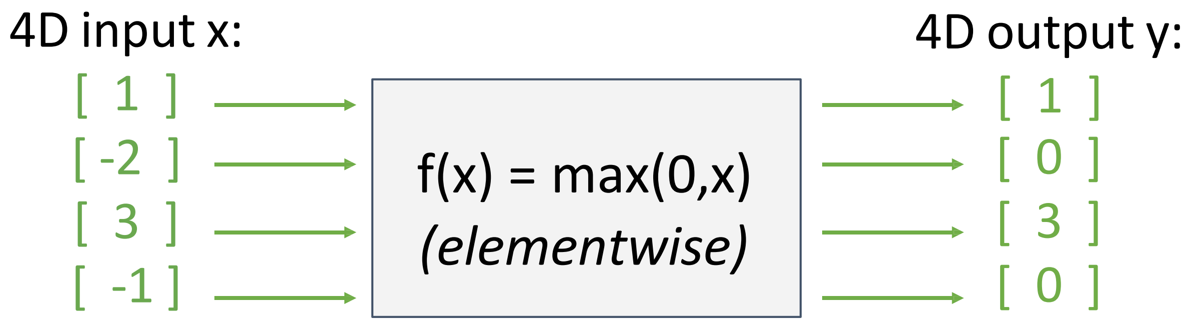
Backprop with Vectors



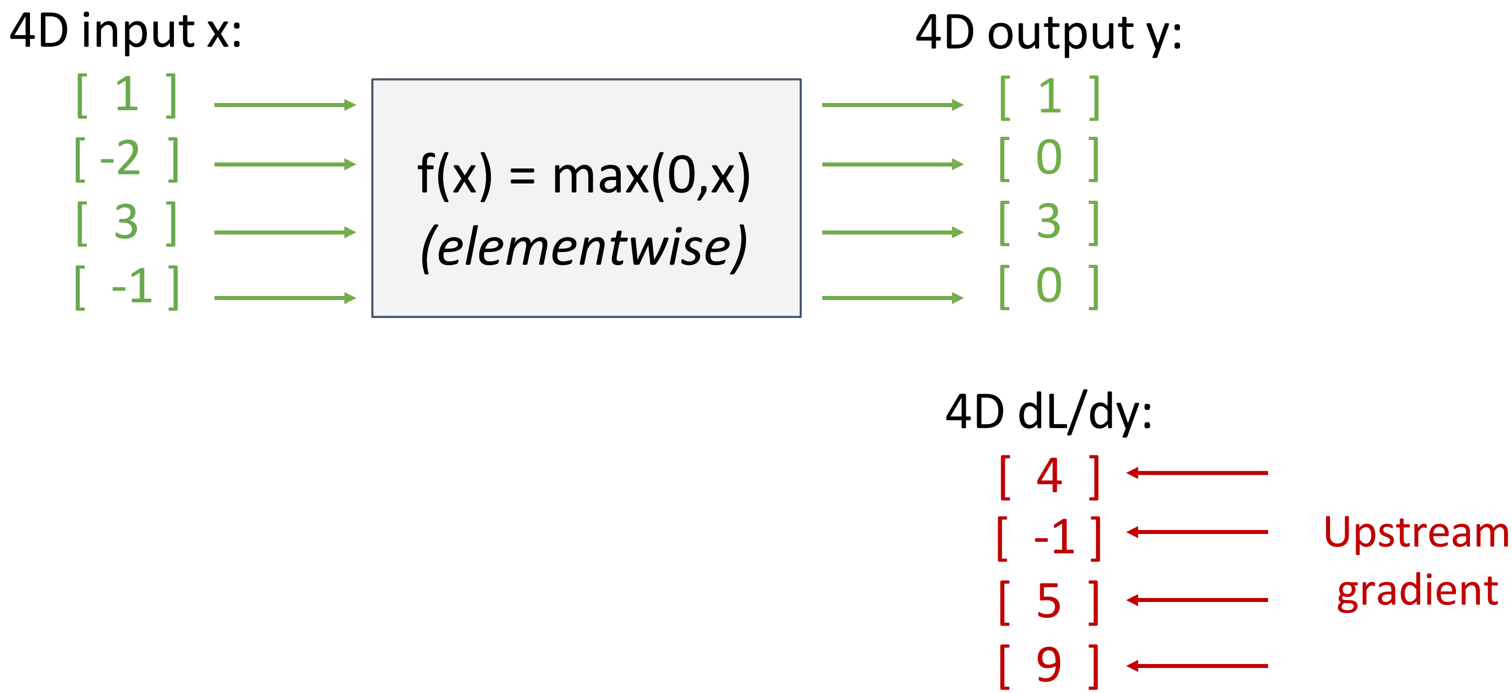
Backprop with Vectors



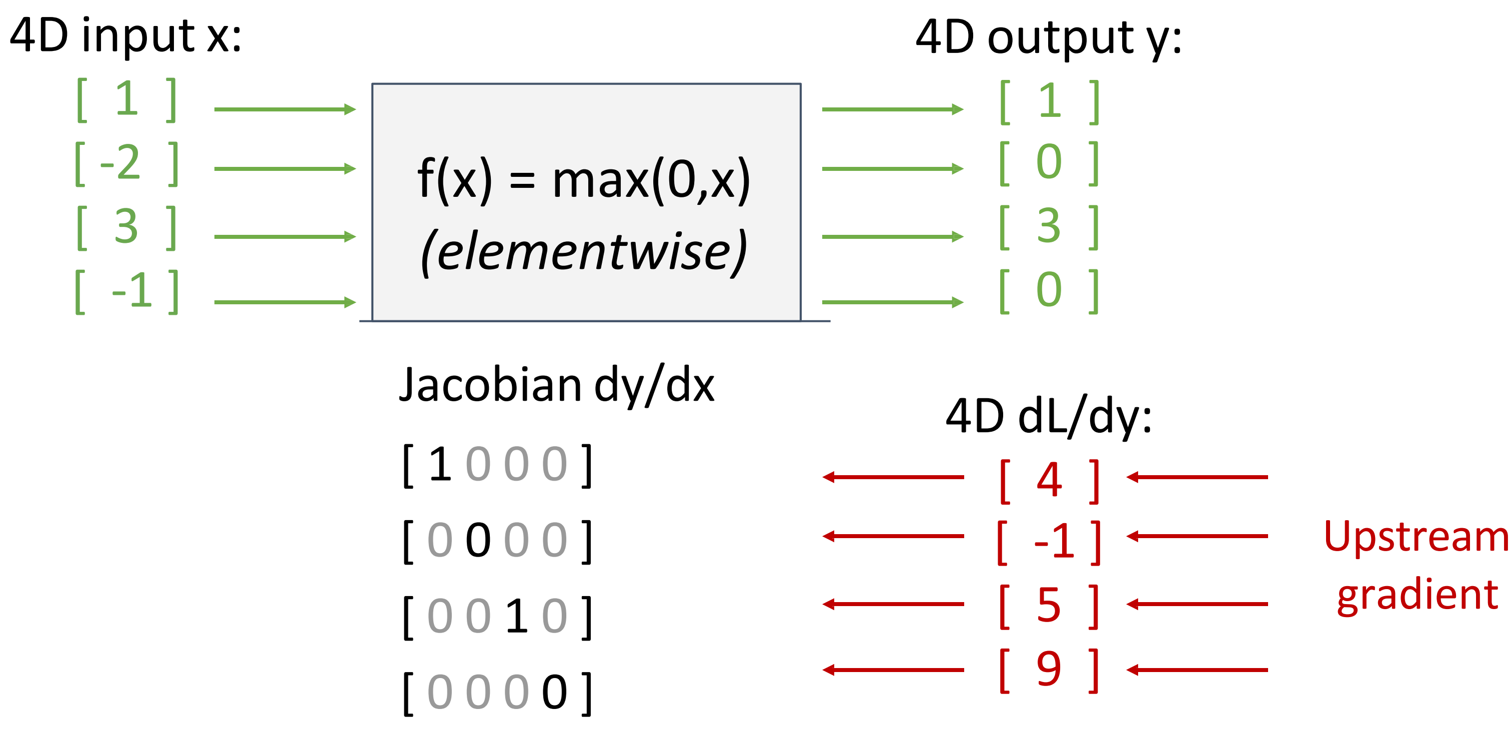
Backprop with Vectors



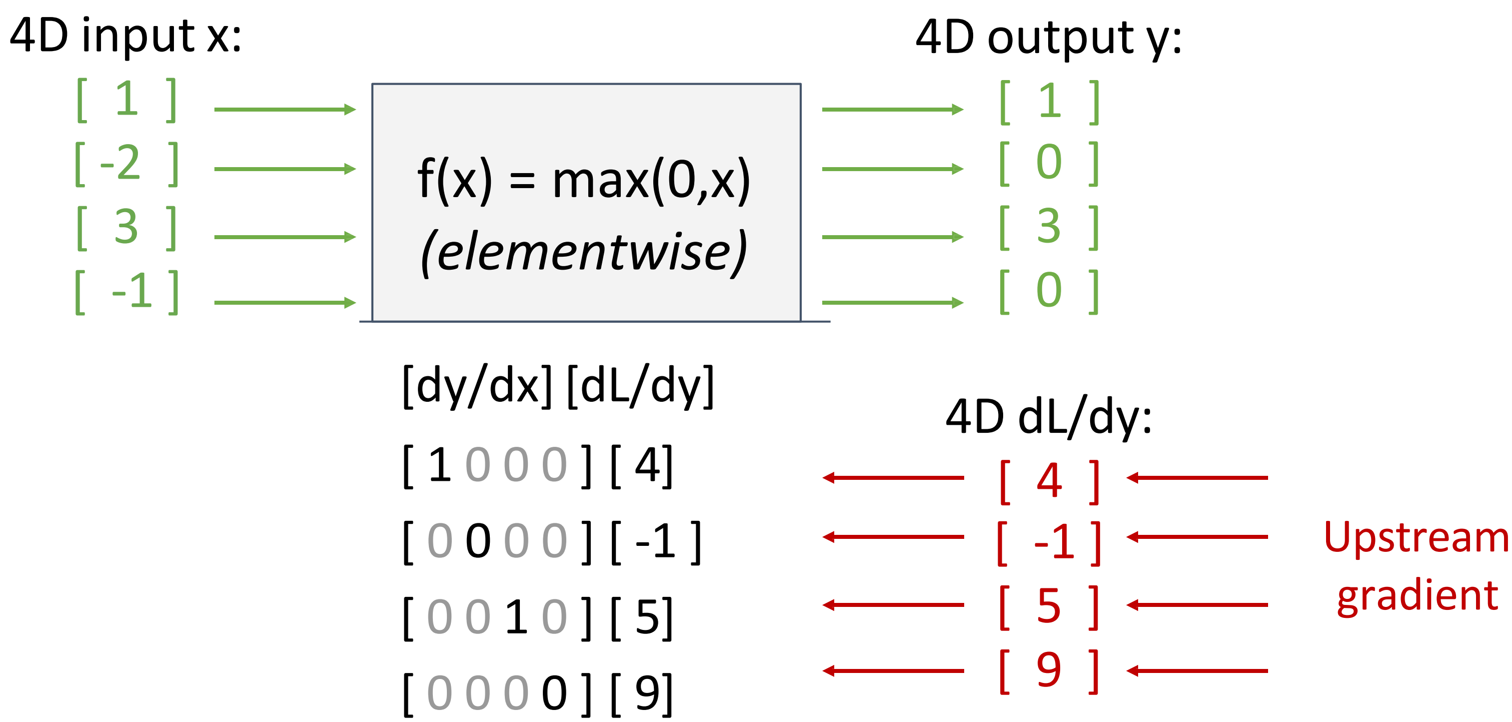
Backprop with Vectors



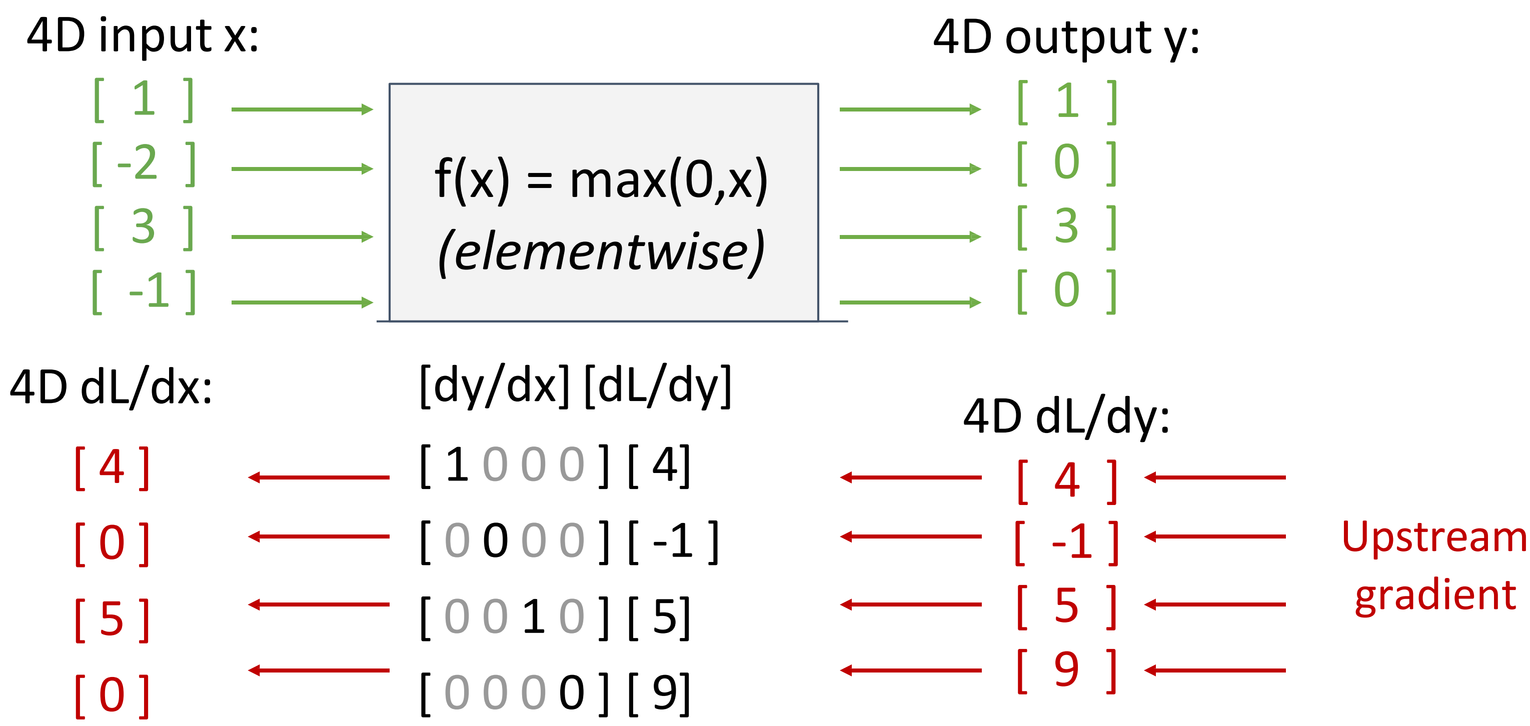
Backprop with Vectors



Backprop with Vectors



Backprop with Vectors



Backprop with Vectors

Jacobian is **sparse**: off-diagonal entries all zero! Never **explicitly** form Jacobian; instead use **implicit** multiplication

4D input x:

[1]
[-2]
[3]
[-1]

f(x) = max(0,x)
(elementwise)

4D output y:

[1]
[0]
[3]
[0]

4D dL/dx:

[4]
[0]
[5]
[0]

[dy/dx] [dL/dy]

[1 0 0 0] [4]
[0 0 0 0] [-1]
[0 0 1 0] [5]
[0 0 0 0] [9]

4D dL/dy:

[4]
[-1]
[5]
[9]

Upstream gradient

Backprop with Vectors

Jacobian is **sparse**: off-diagonal entries all zero! Never **explicitly** form Jacobian; instead use **implicit** multiplication

4D input x:

[1]
[-2]
[3]
[-1]

f(x) = max(0,x)
(elementwise)

4D output y:

[1]
[0]
[3]
[0]

4D dL/dx:

[4]
[0]
[5]
[0]

[dy/dx] [dL/dy]

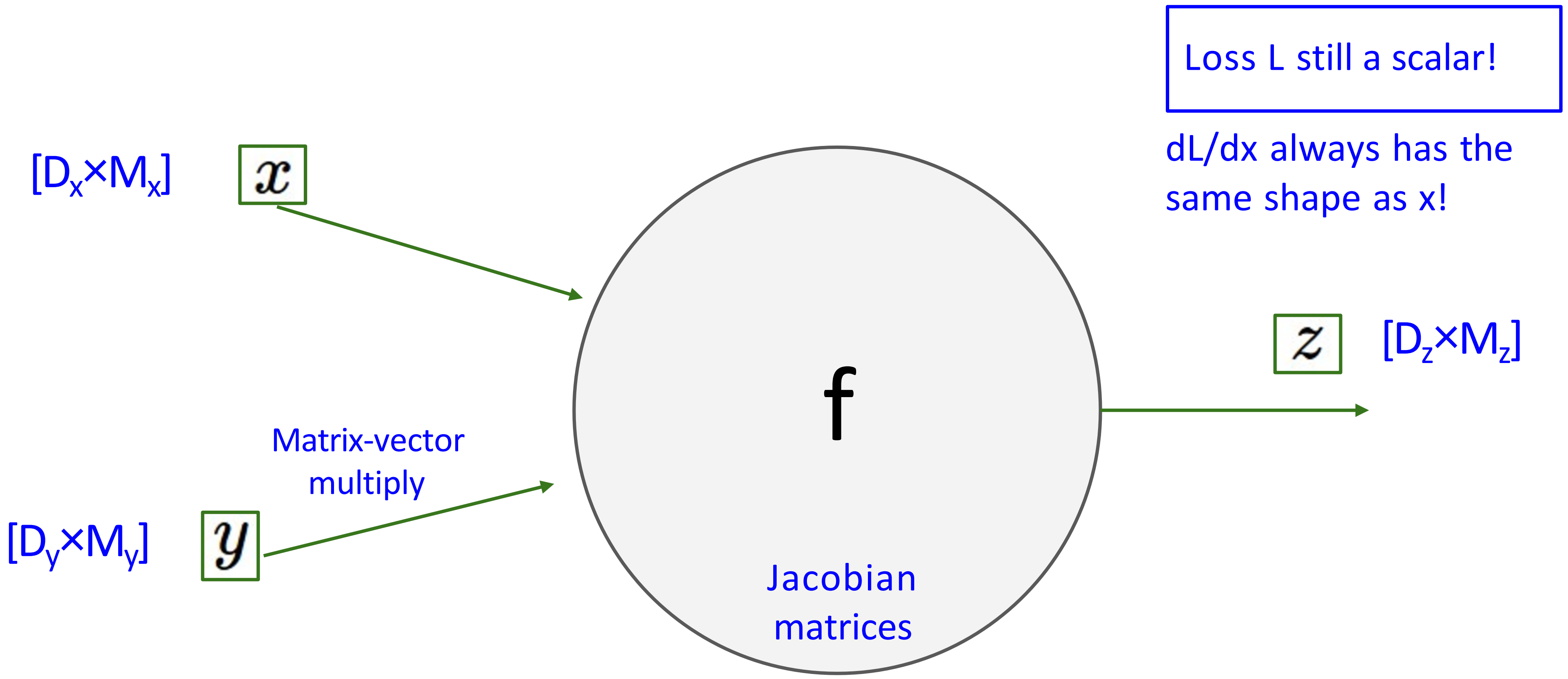
$$\left(\frac{\partial L}{\partial x}\right)_i = \begin{cases} \left(\frac{\partial L}{\partial y}\right)_i & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

4D dL/dy:

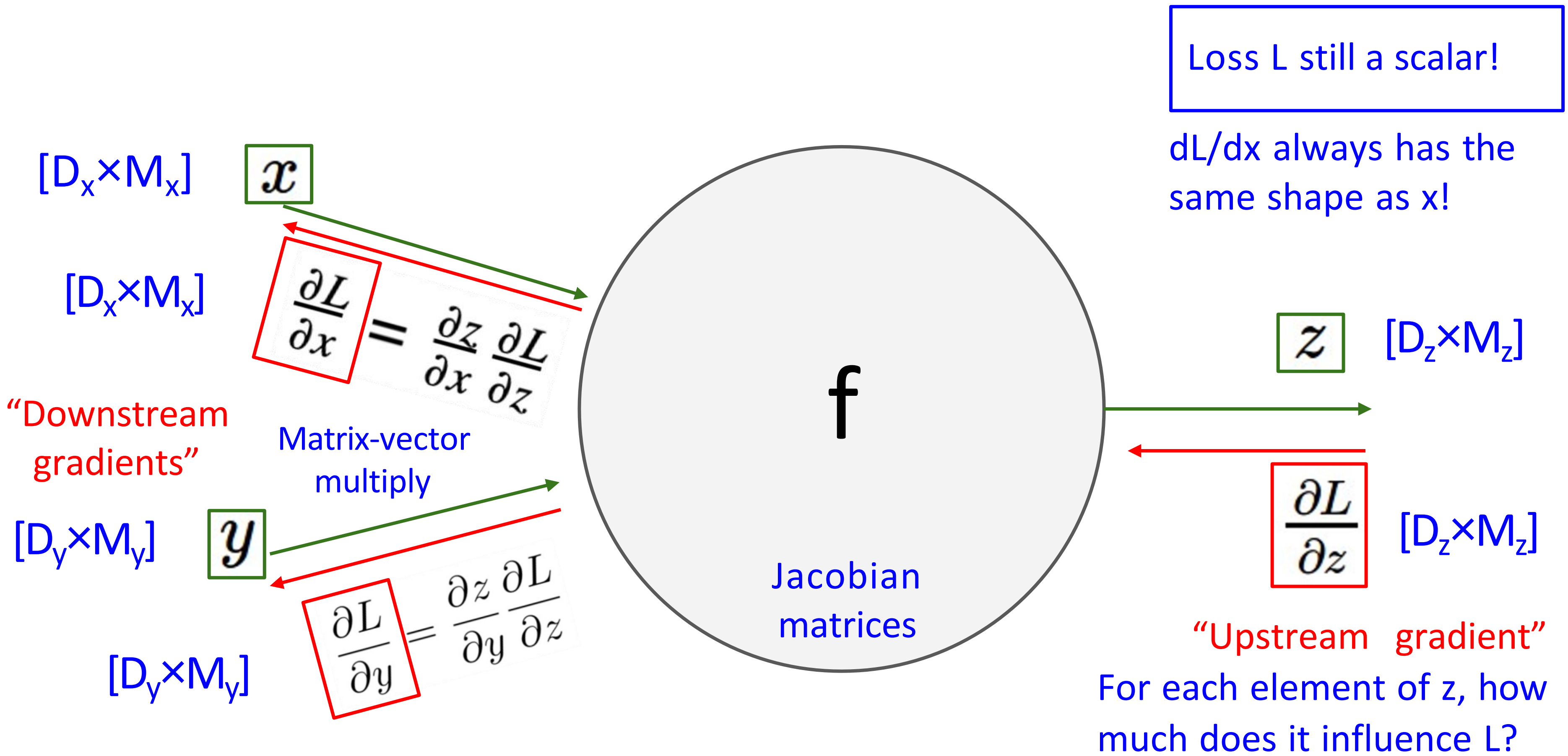
[4]
[-1]
[5]
[9]

Upstream gradient

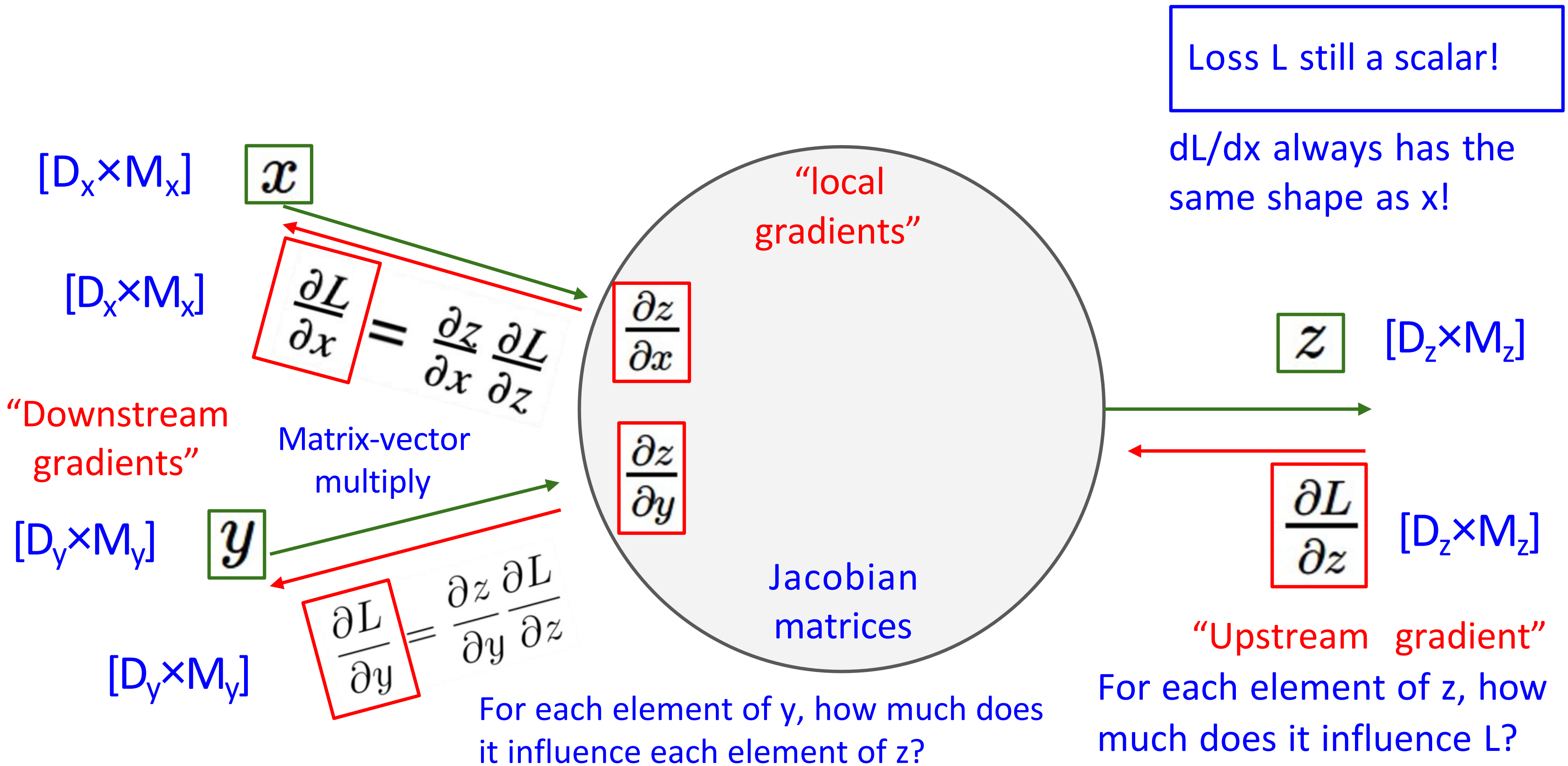
Backprop with Matrices (or Tensors):



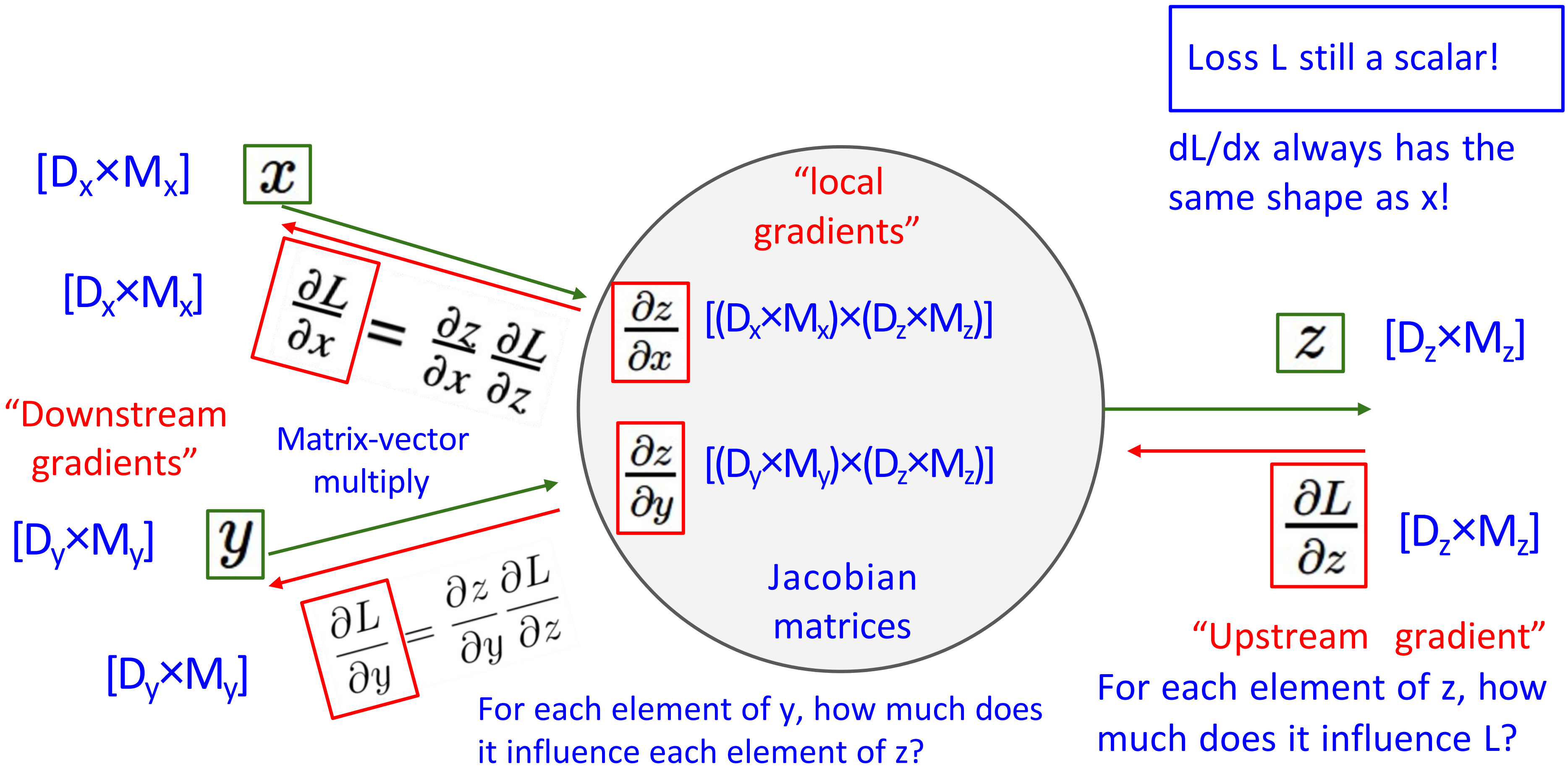
Backprop with Matrices (or Tensors):



Backprop with Matrices (or Tensors):



Backprop with Matrices (or Tensors):



Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Jacobians:

dy/dx:

[(N×D)×(N×M)]

dy/dw: [(D×M)×(N×M)]

For a neural net we may have

N=64, D=M=4096

Each Jacobian takes 256 GB of memory! Must work with them implicitly!

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} \boxed{-1} & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$dy_{1,1}/dx_{1,1}$ $\begin{bmatrix} \boxed{?} & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply $y = xw$

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} \boxed{-1} & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$dy_{1,1}/dx_{1,1}$ $\begin{bmatrix} \boxed{?} & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$

$y_{1,1} = x_{1,1}w_{1,1} + x_{1,2}w_{2,1} + x_{1,3}w_{3,1}$

$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$

w: [D×M]

$\begin{bmatrix} \boxed{3} & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$\begin{bmatrix} \boxed{-1} & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$

dL/dy: [N×M]

$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$

dL/dx:

$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$

Local Gradient Slice:

$dy/dx_{1,1}$

$dy_{1,1}/dx_{1,1}$ $\begin{bmatrix} \boxed{3} & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$

$y_{1,1} = x_{1,1}\boxed{w_{1,1}} + x_{1,2}w_{2,1} + x_{1,3}w_{3,1}$

$\Rightarrow dy_{1,1}/dx_{1,1} = \boxed{w_{1,1}}$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

[2 1 -3]
[-3 4 2]

w: [D×M]

[3 2 1 -1]
[2 1 3 2]
[3 2 1 -2]

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

[-1 -1 2 6]
[5 2 11 7]

dL/dy: [N×M]

[2 3 -3 9]
[-8 1 4 6]

dL/dx:

[? ? ?]
[? ? ?]

$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$

Local Gradient Slice:

$dy/dx_{1,1}$

$dy_{1,1}/dx_{1,1}$

[3 ? ? ?]
[? ? ? ?]

$y_{1,1} = x_{1,1}\boxed{w_{1,1}} + x_{1,2}w_{2,1} + x_{1,3}w_{3,1}$

$\Rightarrow dy_{1,1}/dx_{1,1} = \boxed{w_{1,1}}$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & \boxed{-1} & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$dy_{1,2}/dx_{1,1}$

$$\begin{bmatrix} 3 & \boxed{?} & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & \boxed{-1} & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{array}{cc} dy_{1,2}/dx_{1,1} & \begin{bmatrix} 3 & \boxed{?} & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \end{array}$$

$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$

$$y_{1,2} = x_{1,1}w_{1,2} + x_{1,2}w_{2,2} + x_{1,3}w_{3,2}$$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & \boxed{2} & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & \boxed{-1} & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{array}{cc} dy_{1,2}/dx_{1,1} & \begin{bmatrix} 3 & \boxed{2} & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \end{array}$$

$y_{1,2} = x_{1,1}\boxed{w_{1,2}} + x_{1,2}w_{2,2} + x_{1,3}w_{3,2}$

$\Rightarrow dy_{1,2}/dx_{1,1} = w_{1,2}$

Example: Matrix Multiplication

x: [N×D]

[$\boxed{2}$ 1 -3]
[-3 4 2]

w: [D×M]

[3 2 $\boxed{1\ -1}$]
[2 1 3 2]
[3 2 1 -2]

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

[-1 -1 2 6]
[5 2 11 7]

dL/dy: [N×M]

[2 3 -3 9]
[-8 1 4 6]

dL/dx:

[$\boxed{?}$? ?]
[? ? ?]

Local Gradient Slice:

$dy/dx_{1,1}$

[3 2 $\boxed{1\ -1}$]
[? ? ? ?]

$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ \boxed{5} & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ \boxed{?} & ? & ? & ? \end{bmatrix}$$

$y_{2,1} = x_{2,1}w_{1,1} + x_{2,2}w_{2,1} + x_{2,3}w_{3,1}$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

[2 1 -3]
[-3 4 2]

w: [D×M]

[3 2 1 -1]
[2 1 3 2]
[3 2 1 -2]

Matrix Multiply $y = xw$

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

[-1 -1 2 6]
[5 2 11 7]

dL/dy: [N×M]

[2 3 -3 9]
[-8 1 4 6]

dL/dx:

[? ? ?]
[? ? ?]

Local Gradient Slice:

$dy/dx_{1,1}$

[3 2 1 -1]
[0 ? ? ?]

$y_{2,1} = x_{2,1}w_{1,1} + x_{2,2}w_{2,1} + x_{2,3}w_{3,1}$

$\Rightarrow dy_{2,1}/dx_{1,1} = 0$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{?} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} \boxed{2} & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} \boxed{3 & 2 & 1 & -1} \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} \boxed{2 & 3 & -3 & 9} \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} \boxed{0} & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{1,1}$

$$\begin{bmatrix} \boxed{3 & 2 & 1 & -1} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$dL/dx_{1,1}$

$= (dy/dx_{1,1}) \cdot (dL/dy)$

$= (w_{1,:}) \cdot (dL/dy_{1,:})$

$= 3*2 + 2*3 + 1*(-3) + (-1)*9 = 0$

Example: Matrix Multiplication

x: [N×D]

[2 1 -3]
[-3 4 $\boxed{2}$]

w: [D×M]

[3 2 1 -1]
[2 1 3 2]
[$\boxed{3\ 2\ 1\ -2}$]

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

[-1 -1 2 6]
[5 2 11 7]

dL/dy: [N×M]

[2 3 -3 9]
[-8 1 4 6]

dL/dx:

[0 ? ?]
[? ? $\boxed{?}$]

Local Gradient Slice:

$dy/dx_{2,3}$

[0 0 0 0]
[3 2 1 -2]

$dL/dx_{2,3}$
$= (dy/dx_{2,3}) \cdot (dL/dy)$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & \boxed{2} \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ \hline 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} 0 & ? & ? \\ ? & ? & \boxed{-30} \end{bmatrix}$$

Local Gradient Slice:

$dy/dx_{2,3}$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

$dL/dx_{2,3}$

$= (dy/dx_{2,3}) \cdot (dL/dy)$

$= (w_{3,:}) \cdot (dL/dy_{2,:})$

$= 3*(-8) + 2*1 + 1*4 + (-2)*6 = -30$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} 0 & 16 & -9 \\ -24 & 9 & -30 \end{bmatrix}$$

$dL/dx_{i,j}$

$= (dy/dx_{i,j}) \cdot (dL/dy)$

$= (w_{j,:}) \cdot (dL/dy_{i,:})$

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} 0 & 16 & -9 \\ -24 & 9 & -30 \end{bmatrix}$$

$$\underbrace{dL/dx}_{[N \times D]} = \underbrace{(dL/dy)}_{[N \times M]}\ \underbrace{w^T}_{[M \times D]}$$

$dL/dx_{i,j}$

$= (dy/dx_{i,j}) \cdot (dL/dy)$

$= (w_{j,:}) \cdot (dL/dy_{i,:})$

Easy way to remember:
It's the only way the
shapes work out!

Example: Matrix Multiplication

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

Matrix Multiply *y = xw*

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

y: [N×M]

$$\begin{bmatrix} -1 & -1 & 2 & 6 \\ 5 & 2 & 11 & 7 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

dL/dx:

$$\begin{bmatrix} 0 & 16 & -9 \\ -24 & 9 & -30 \end{bmatrix}$$

$$\underbrace{dL/dx}_{[N \times D]} = \underbrace{(dL/dy)}_{[N \times M]}\ \underbrace{w^T}_{[M \times D]}$$

$$\underbrace{dL/dw}_{[D \times M]} = \underbrace{x^T}_{[D \times N]}\ \underbrace{(dL/dy)}_{[N \times M]}$$

Easy way to remember: It's the only way the shapes work out!

Backpropagation: Another View

$$
\begin{tikzcd}
x_0 \arrow[r, "f_1"] & x_1 \arrow[r, "f_2"] & x_2 \arrow[r, "f_3"] & x_3 \arrow[r, "f_4"] & L \\
D_0 & D_1 & D_2 & D_3 & \text{scalar}
\end{tikzcd}
$$

Chain rule $$\frac{\partial L}{\partial x_0} = \left(\frac{\partial x_1}{\partial x_0}\right) \left(\frac{\partial x_2}{\partial x_1}\right) \left(\frac{\partial x_3}{\partial x_2}\right) \left(\frac{\partial L}{\partial x_3}\right)$$

Backprop: Higher-Order Derivatives

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} L$$

D_0 $\quad$ D_1 $\quad$ scalar

$$\frac{\partial^2 L}{\partial x_0^2}$$

$D_0 \times D_0$

Hessian matrix
H of second derivatives.

Backprop: Higher-Order Derivatives

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} L$$

D_0 D_1 scalar

$$\frac{\partial^2 L}{\partial x_0^2}$$

Hessian matrix
H of second derivatives.

$D_0 \times D_0$

Hessian / vector multiply

$$\frac{\partial^2 L}{\partial x_0^2} v$$

$D_0 \times D_0$ D_0

Backprop: Higher-Order Derivatives

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} L$$

D_0 D_1 scalar

Hessian / vector multiply

$$\frac{\partial^2 L}{\partial x_0^2}$$

Hessian matrix
H of second derivatives.

D_0 x D_0

$$\frac{\partial^2 L}{\partial x_0^2} v = \frac{\partial}{\partial x_0}\left[\frac{\partial L}{\partial x_0} \cdot v\right]$$

(if v doesn't depend on x_0)

D_0 x D_0 D_0

Backprop: Higher-Order Derivatives

$$\underset{D_0}{x_0} \xrightarrow{f_1} \underset{D_1}{x_1} \xrightarrow{f_2} \underset{\text{scalar}}{L} \xrightarrow{f'_2} \underset{D_1}{dL/dx_1} \xrightarrow{f'_1} \underset{D_0}{dL/dx_0} \xrightarrow{\cdot v} \underset{\text{scalar}}{(dL/dx_0) \cdot v}$$

$$\underset{D_0 \times D_0}{\frac{\partial^2 L}{\partial x_0^2}}$$

Hessian matrix
H of second derivatives.

Hessian / vector multiply

$$\underset{D_0 \times D_0}{\frac{\partial^2 L}{\partial x_0^2}} \underset{D_0}{v} = \frac{\partial}{\partial x_0}\left[\frac{\partial L}{\partial x_0} \cdot v\right]$$

(if v doesn't depend on x_0)

Backprop: Higher-Order Derivatives

$$\underset{D_0}{x_0} \xrightarrow{f_1} \underset{D_1}{x_1} \xrightarrow{f_2} \underset{\text{scalar}}{L} \xrightarrow{f'_2} \underset{D_1}{dL/dx_1} \xrightarrow{f'_1} \underset{D_0}{dL/dx_0} \xrightarrow{\cdot v} \underset{\text{scalar}}{(dL/dx_0) \cdot v}$$

Backprop!

$$\underset{D_0 \times D_0}{\frac{\partial^2 L}{\partial x_0^2}}$$

Hessian matrix H of second derivatives.

Hessian / vector multiply

$$\underset{D_0 \times D_0}{\frac{\partial^2 L}{\partial x_0^2}} \underset{D_0}{v} = \frac{\partial}{\partial x_0}\left[\frac{\partial L}{\partial x_0} \cdot v\right]$$

(if v doesn't depend on x_0)

Backprop: Higher-Order Derivatives

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} L \xrightarrow{f'_2} dL/dx_1 \xrightarrow{f'_1} dL/dx_0 \xrightarrow{\cdot v} (dL/dx_0) \cdot v$$

D_0 | D_1 | scalar | D_1 | D_0 | scalar

Backprop!

This is implemented in PyTorch / Tensorflow!

$$\frac{\partial^2 L}{\partial x_0^2}$$

Hessian matrix H of second derivatives.

$D_0 \times D_0$

Hessian / vector multiply

$$\frac{\partial^2 L}{\partial x_0^2} v = \frac{\partial}{\partial x_0}\left[\frac{\partial L}{\partial x_0} \cdot v\right]$$

(if v doesn't depend on x_0)

$D_0 \times D_0$ $\quad D_0$

Backprop: Higher-Order Derivatives

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} L \xrightarrow{f'_2} dL/dx_1 \xrightarrow{f'_1} dL/dx_0 \xrightarrow{\cdot v} (dL/dx_0) \cdot v$$

D_0 D_1 scalar D_1 D_0 scalar

Backprop!

This is implemented in PyTorch / Tensorflow!

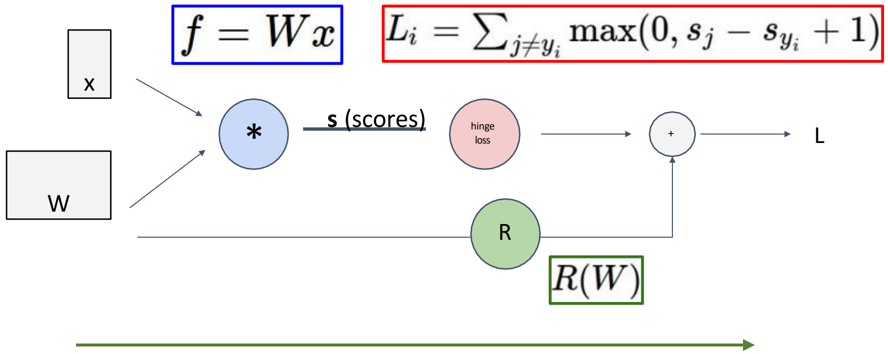
Example: Regularization to penalize the norm of the gradient

$$R(W) = \left\| \frac{\partial L}{\partial W} \right\|_2^2 = \left(\frac{\partial L}{\partial W} \right) \cdot \left(\frac{\partial L}{\partial W} \right) \qquad \frac{\partial}{\partial x_0} \left[R(W) \right] = 2 \left(\frac{\partial^2 L}{\partial x_0^2} \right) \left(\frac{\partial L}{\partial x_0} \right)$$

Gulrajani et al, "Improved Training of Wasserstein GANs", NeurIPS 2017

Summary

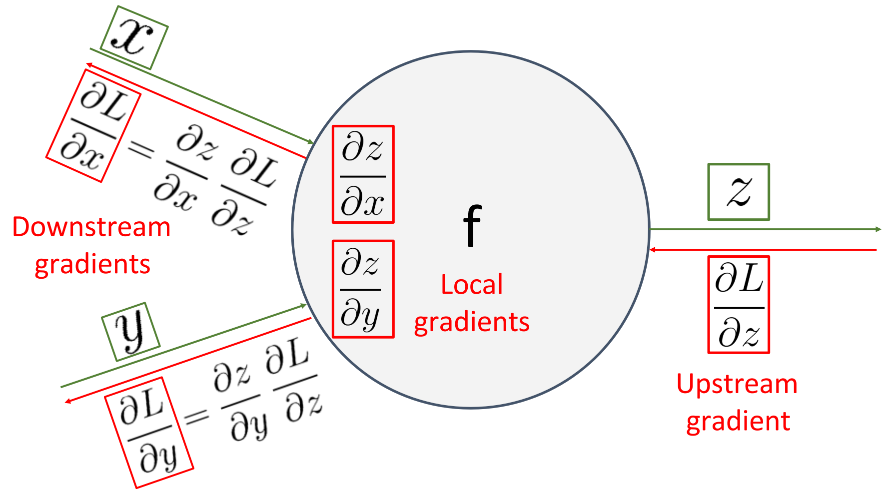
Represent complex expressions as **computational graphs**



Forward pass computes outputs

Backward pass computes gradients

During the backward pass, each node in the graph receives **upstream gradients** and multiplies them by **local gradients** to compute **downstream gradients**



Summary

Backprop can be implemented with "flat" code where the backward pass looks like forward pass reversed.

```
def f(w0, x0, w1, x1, w2):
  s0 = w0 * x0
  s1 = w1 * x1
  s2 = s0 + s1
  s3 = s2 + w2
  L = sigmoid(s3)

  grad_L = 1.0
  grad_s3 = grad_L * (1 - L) * L
  grad_w2 = grad_s3
  grad_s2 = grad_s3
  grad_s0 = grad_s2
  grad_s1 = grad_s2
  grad_w1 = grad_s1 * x1
  grad_x1 = grad_s1 * w1
  grad_w0 = grad_s0 * x0
  grad_x0 = grad_s0 * w0
```

Backprop can be implemented with a modular API, as a set of paired forward/backward functions (We will do this on A3!)

```
class Multiply(torch.autograd.Function):
  @staticmethod
  def forward(ctx, x, y):
    ctx.save_for_backward(x, y)
    z = x * y
    return z
  @staticmethod
  def backward(ctx, grad_z):
    x, y = ctx.saved_tensors
    grad_x = y * grad_z   # dz/dx * dL/dz
    grad_y = x * grad_z   # dz/dy * dL/dz
    return grad_x, grad_y
```