CS 446/ECE 449: Machine Learning

Shenlong Wang

University of Illinois at Urbana-Champaign, 2024

L9: Deep Neural Networks

Goals of this lecture

- Understanding forward and backward pass
- Learning about backpropagation

Reading material

- I. Goodfellow et al.; Deep Learning; Chapters 6-9

Goals of this lecture

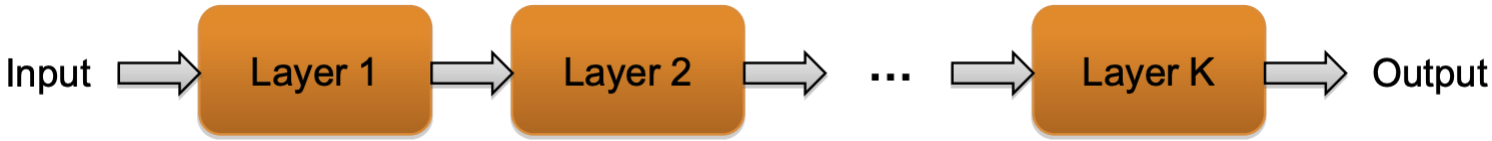
- Stochastic Gradient Decent (SGD)
- Understanding forward and backward pass
- Learning about backpropagation

Reading material

- I. Goodfellow et al.; Deep Learning; Chapters 6-9

Recall: Multi-Layer Neural Networks

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



Neural networks as functions.

A linear predictor (**one layer network**) has the form

$$\boldsymbol{x} \mapsto \boldsymbol{w}^\top \boldsymbol{x}.$$

A **two layer network** has the form

$$\boldsymbol{x} \mapsto \boldsymbol{w}^\top \sigma_1 \left(\boldsymbol{A}_1 \boldsymbol{x} + \mathbf{b}_1 \right).$$

Iterating, a **multi-layer network** has the form

$$\boldsymbol{x} \mapsto \boldsymbol{w}^\top \sigma_1 \left(\boldsymbol{A}_1 \sigma_2 \left(\cdots \boldsymbol{A}_{L-2} \sigma_{L-1} \left(\boldsymbol{A}_{L-1} \boldsymbol{x} + \mathbf{b}_{L-1} \right) + \mathbf{b}_{L-2} \cdots \right) + \mathbf{b}_1 \right).$$

ERM now takes the form

$$\underset{\boldsymbol{w}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_{L-1}, \mathbf{b}_1, \ldots, \mathbf{b}_{L-1}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell \left(y^{(i)} \boldsymbol{w}^\top \sigma_1 \left(\cdots \sigma_{L-1} \left(\boldsymbol{A}_{L-1} \mathbf{x}^{(i)} + \mathbf{b}_{L-1} \right) \cdots \right) \right).$$

How?

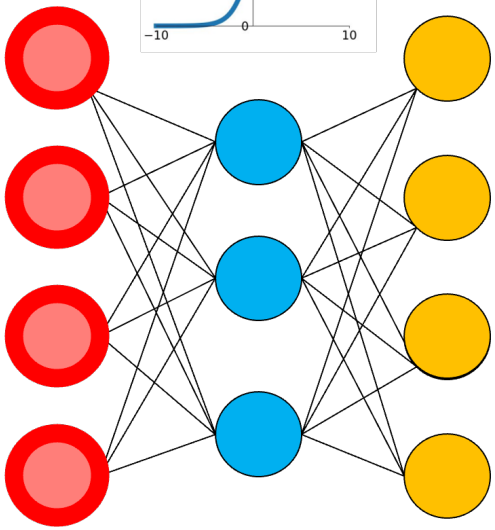
Back-propagation

Consider function: $y = f(g(x))$, what is the gradient of $\frac{\partial y}{\partial x}$?

Chain rule: Let $y = f(z), z = g(x)$, then using chain rule we have:

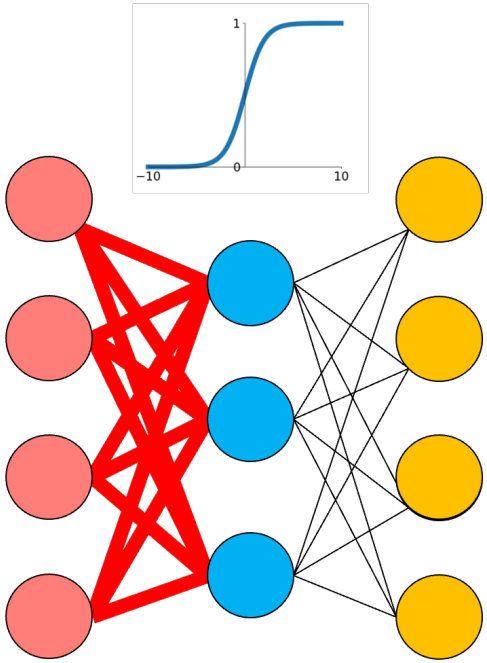
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial x}$$

Forward pass



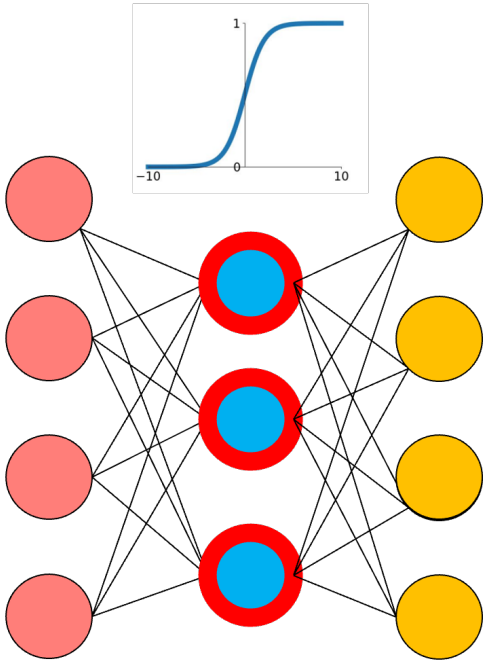
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Forward pass



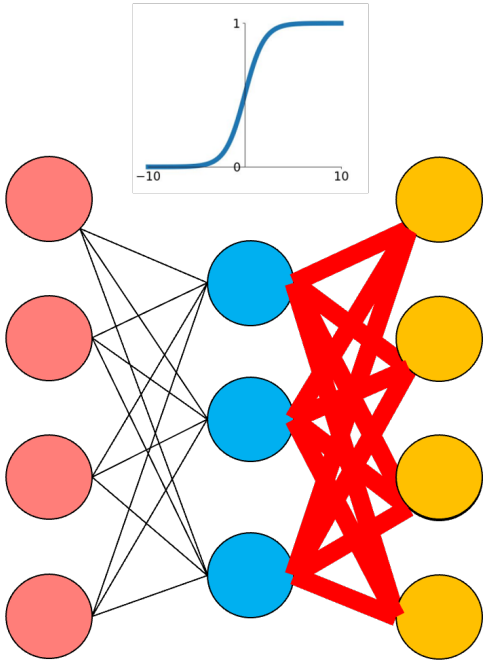
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Forward pass



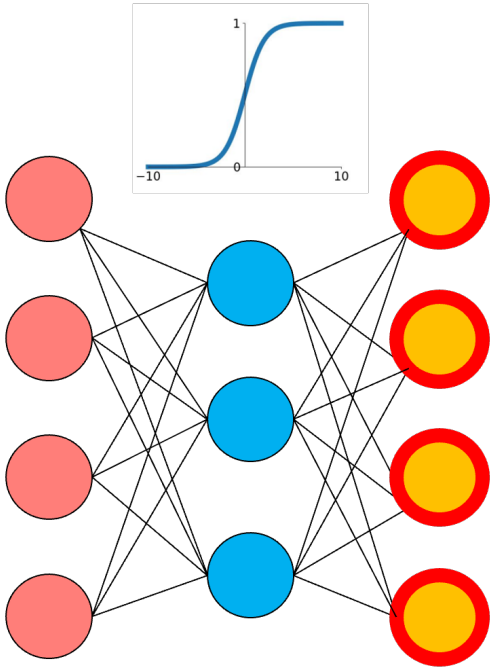
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Forward pass



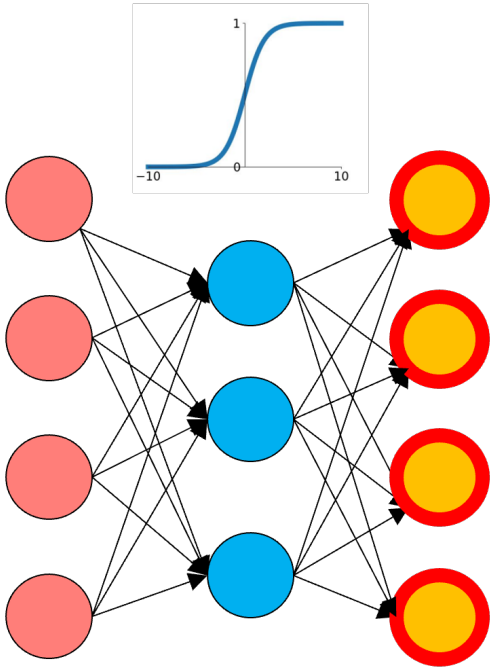
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Forward pass



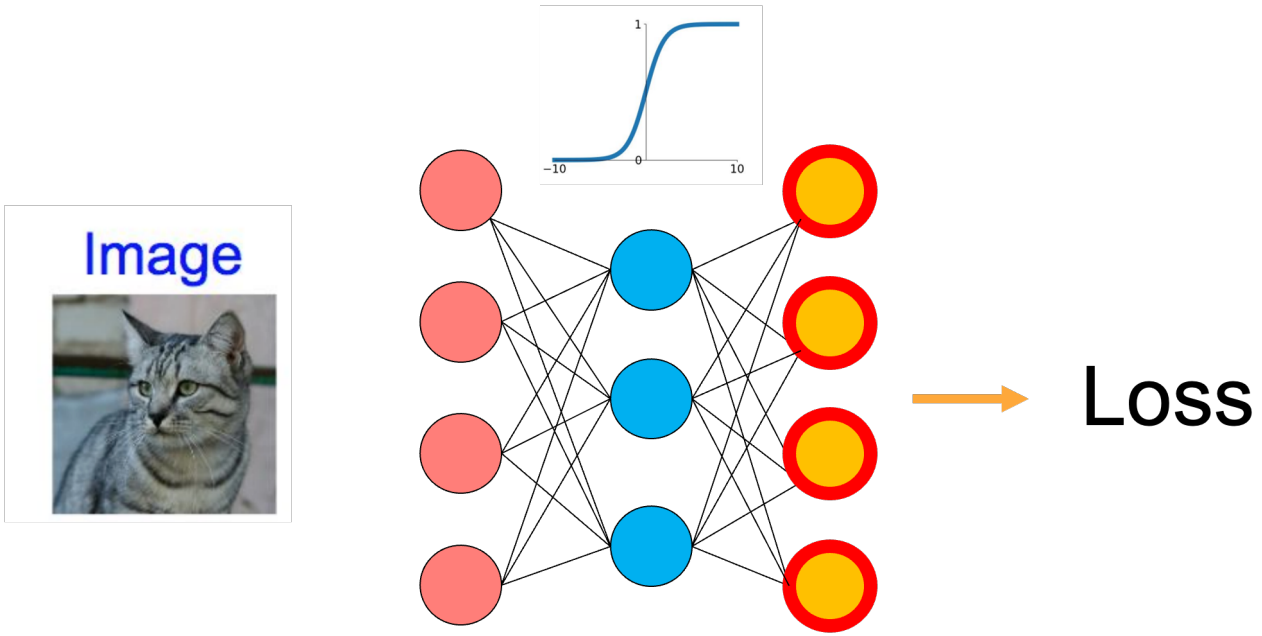
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Forward pass



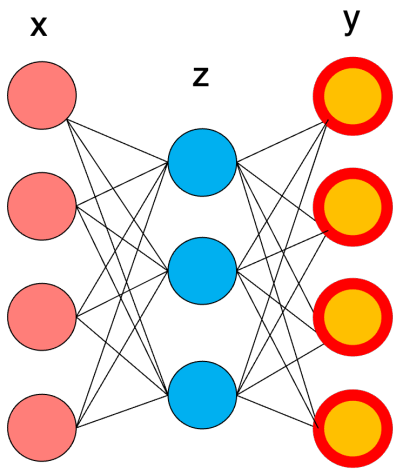
$$y = W_2\sigma(W_1x+b_1)+b_2$$

Backward pass



$$y = W_2\sigma(W_1x+b_1)+b_2$$

Backward pass



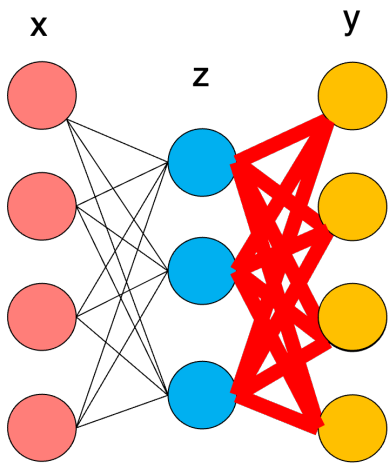
L

Loss

$y = W_2\sigma(W_1x+b_1)+b_2$

$$\frac{\partial L}{\partial y}$$

Backward pass

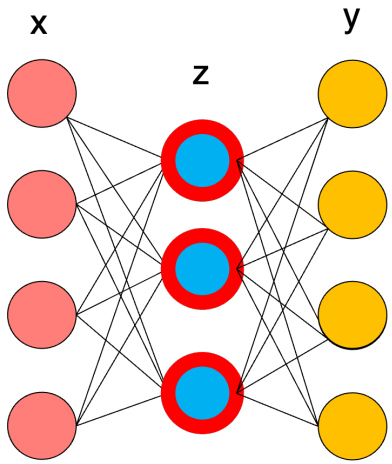


Loss

$y = W_2\sigma(W_1x+b_1)+b_2$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W_2}$$

Backward pass

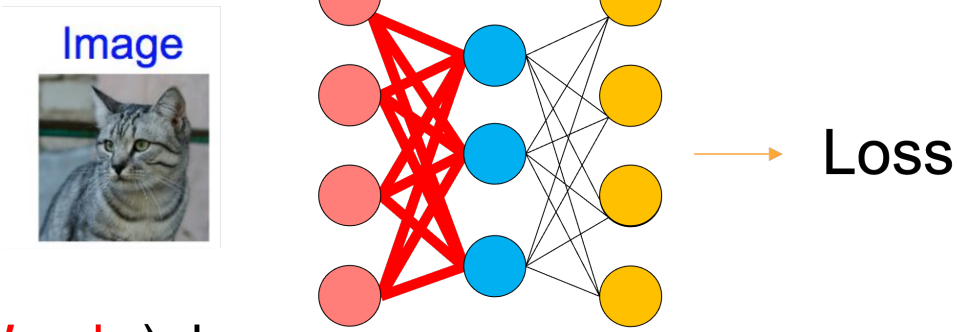


Loss

$y = W_2\sigma(W_1x+b_1)+b_2$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W_2} \qquad \boxed{\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}}$$

Backward pass



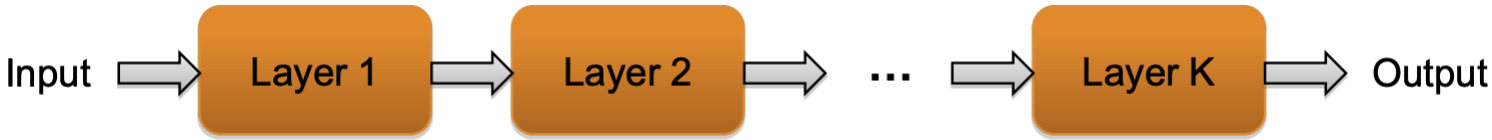
$y = W_2\sigma(W_1x+b_1)+b_2$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W_2}$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial W_1}$$

Recall: Multi-Layer Neural Networks

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



How to implement this gradient computation?

Example:

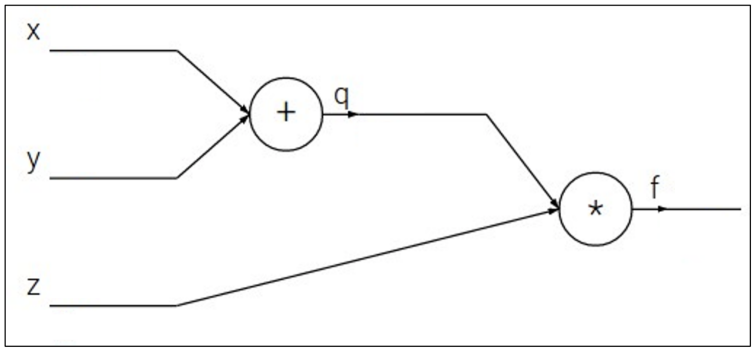
Derive each $\frac{\partial L}{\partial w}$ using chain rule on paper? —Bad idea! Why?

- Very tedious; lots of matrix calculus
- What if we want to change loss? Need to re-derive from scratch. Not modular

Better idea: Computation graph.

Backpropagation: Simple Example

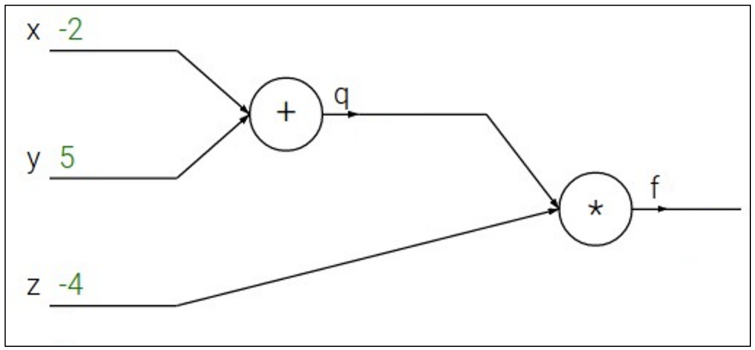
$$f(x, y, z) = (x + y)z$$



Backpropagation: Simple Example

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

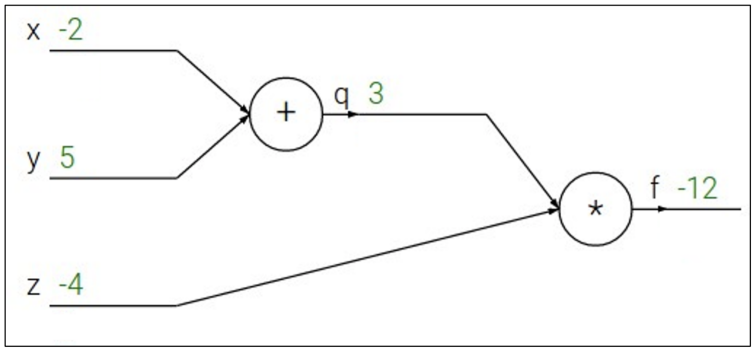
e.g. x = -2, y = 5, z = -4



Backpropagation: Simple Example

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

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



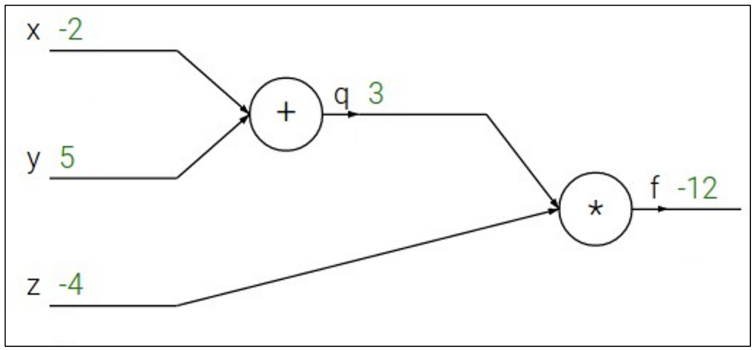
1. Forward pass: Compute outputs

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

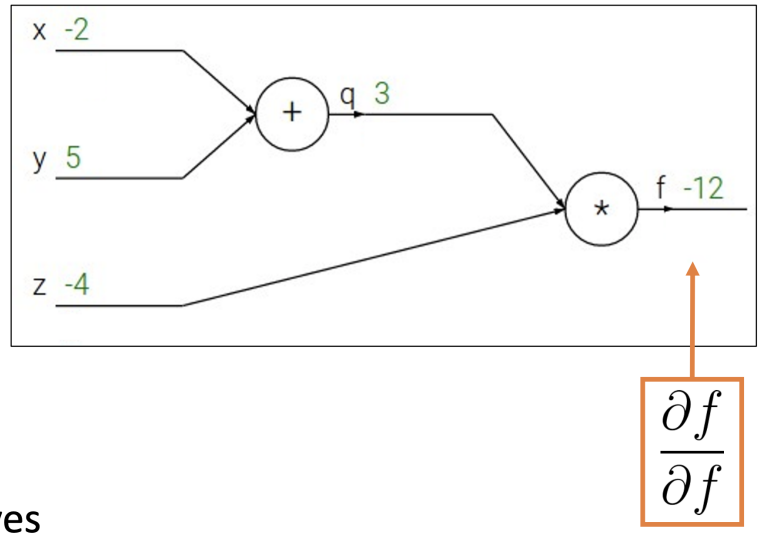
2. Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

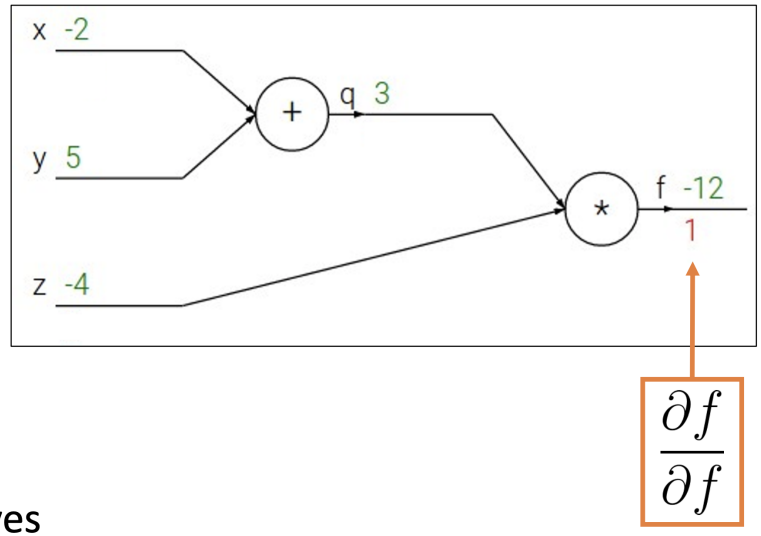
2. Backward pass: Compute derivatives

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

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

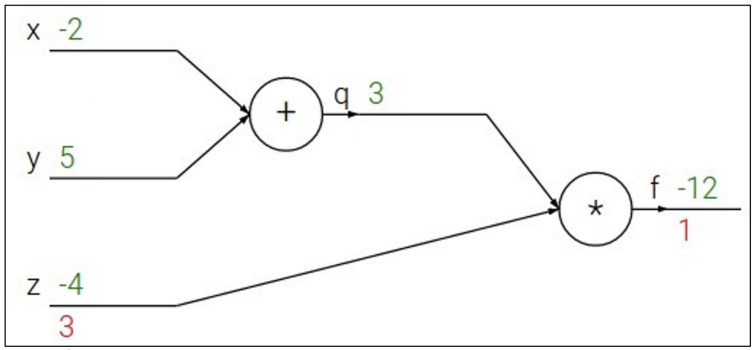
2. Backward pass: Compute derivatives

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

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

2. Backward pass: Compute derivatives

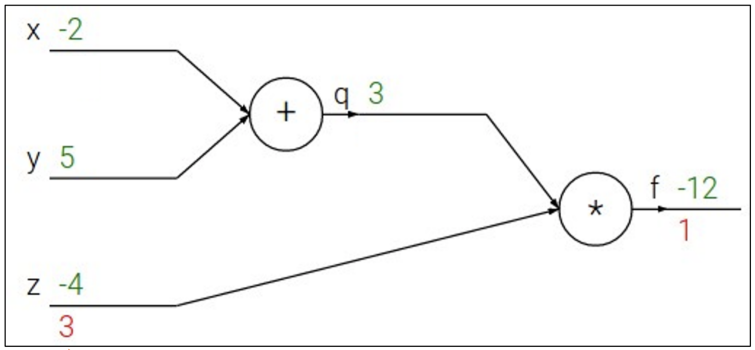
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

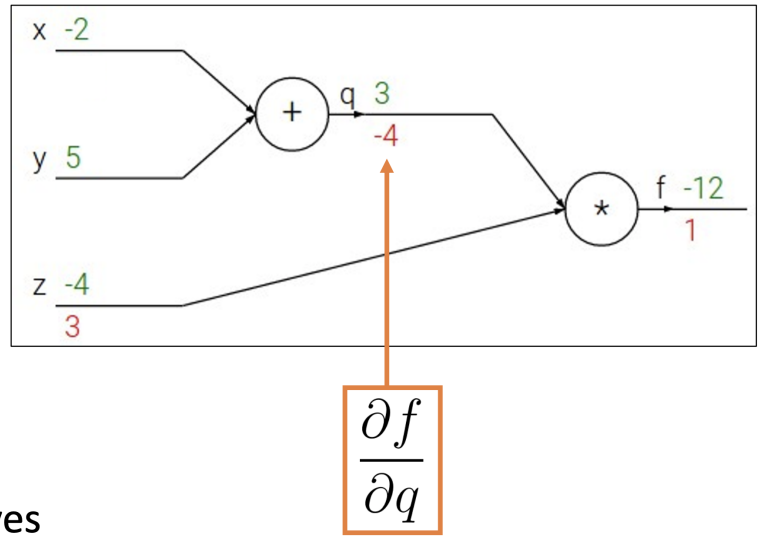
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

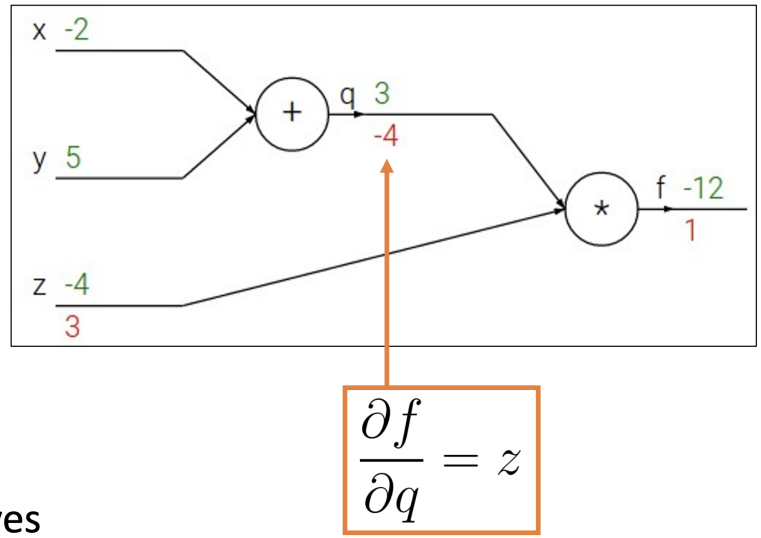
2. Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

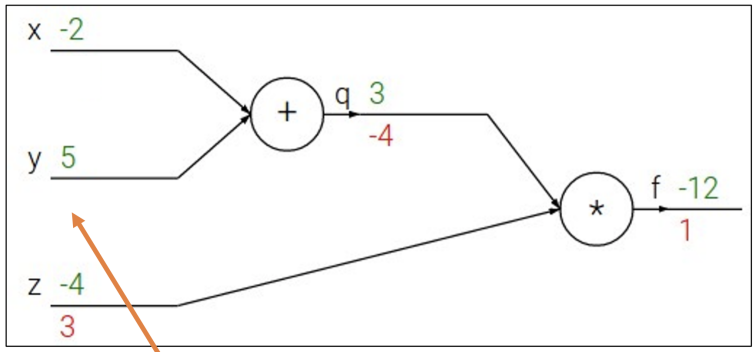
2. Backward pass: Compute derivatives

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

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

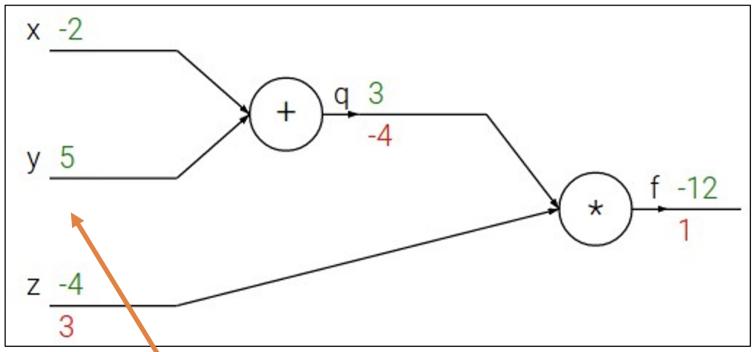
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

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

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

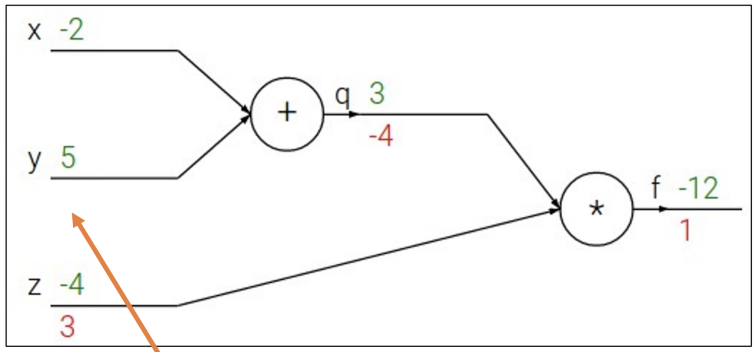
Chain Rule

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

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

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

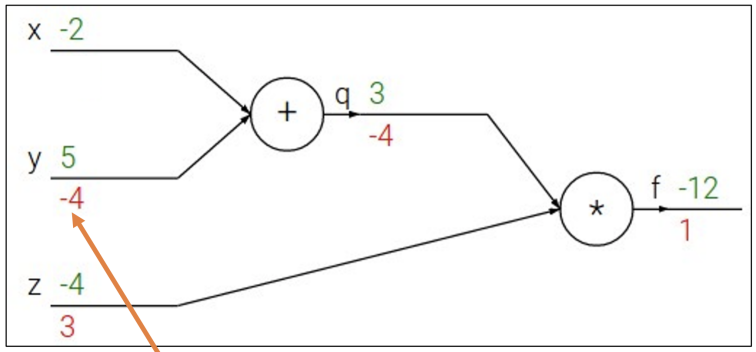
$$\frac{\partial q}{\partial y} = 1$$

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

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

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

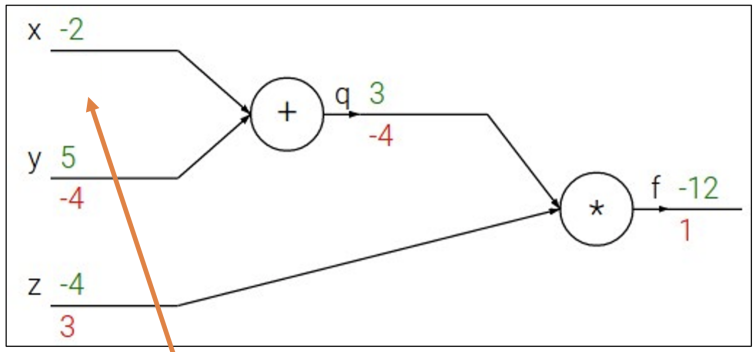
$$\frac{\partial q}{\partial y} = 1$$

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

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

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

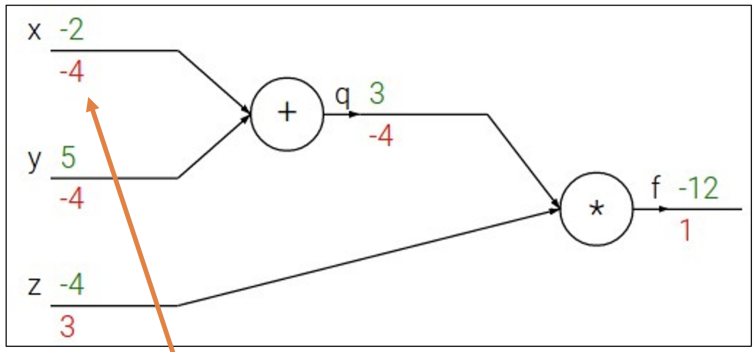
$$\frac{\partial q}{\partial x} = 1$$

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

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

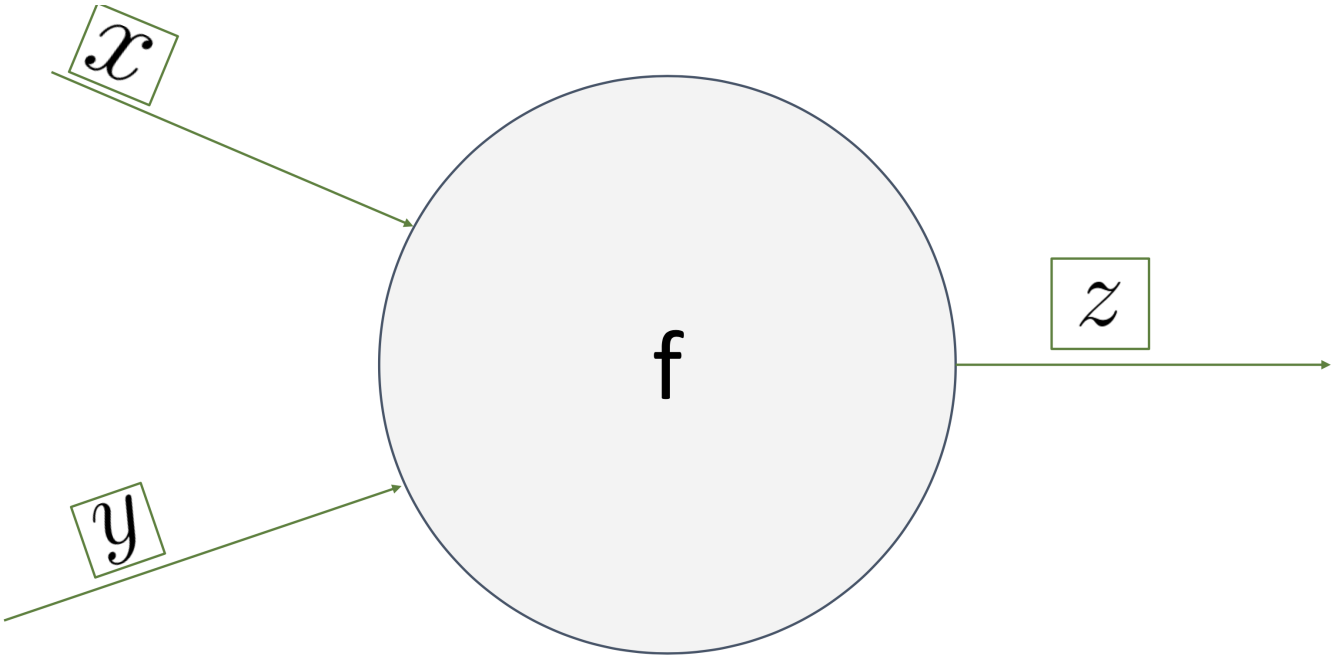
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

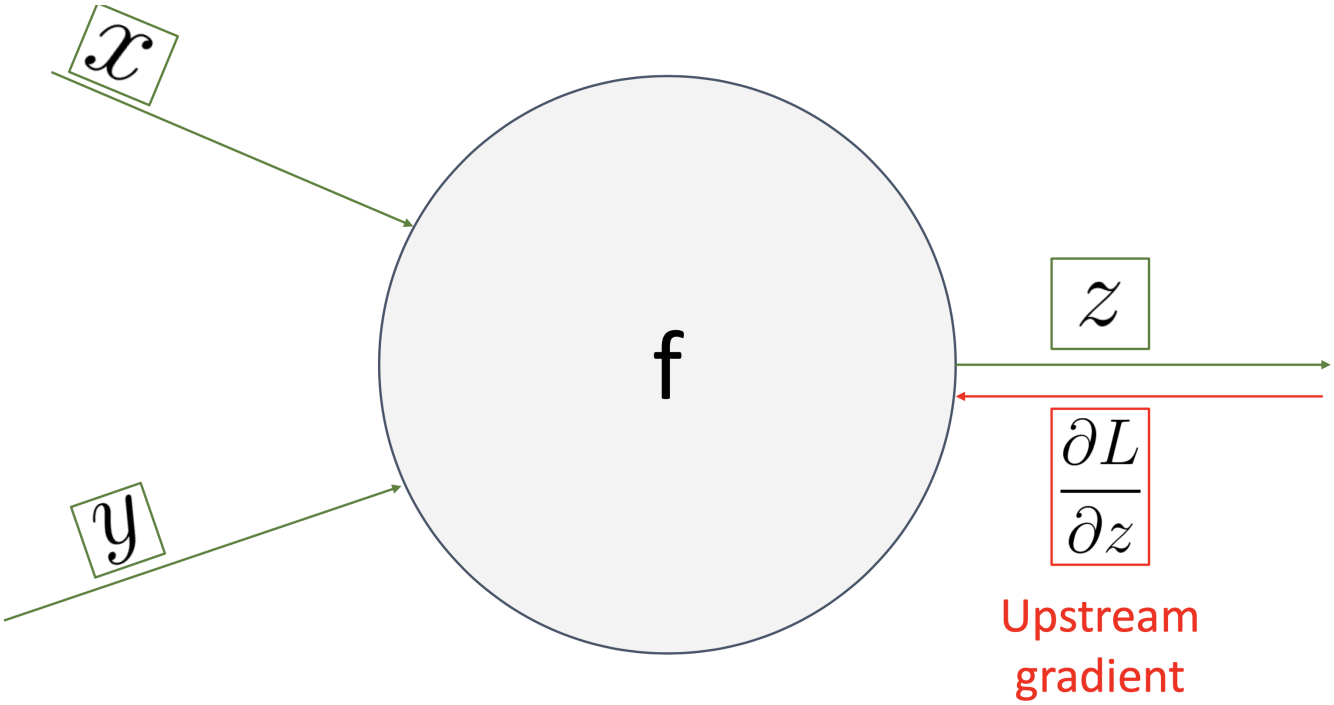
Chain Rule

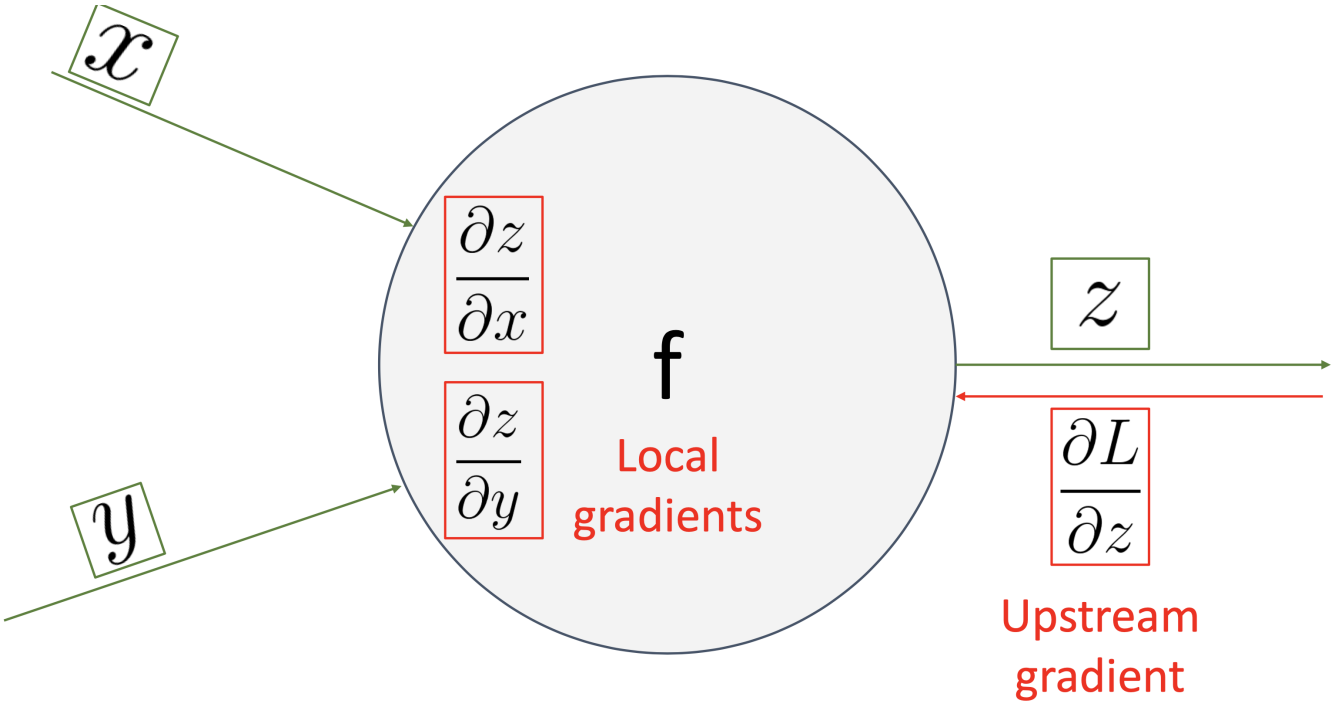
$$\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}$$

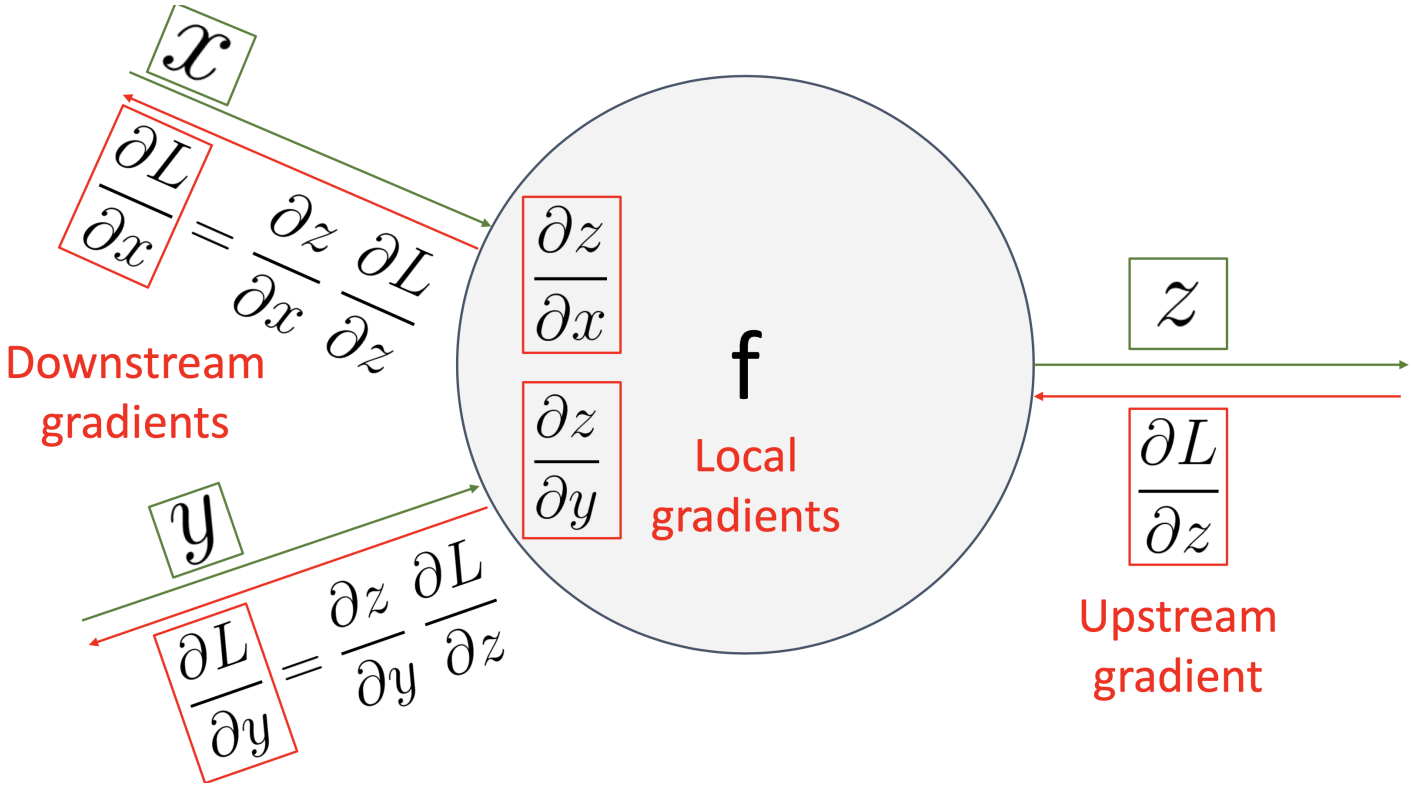
$$\frac{\partial q}{\partial x} = 1$$

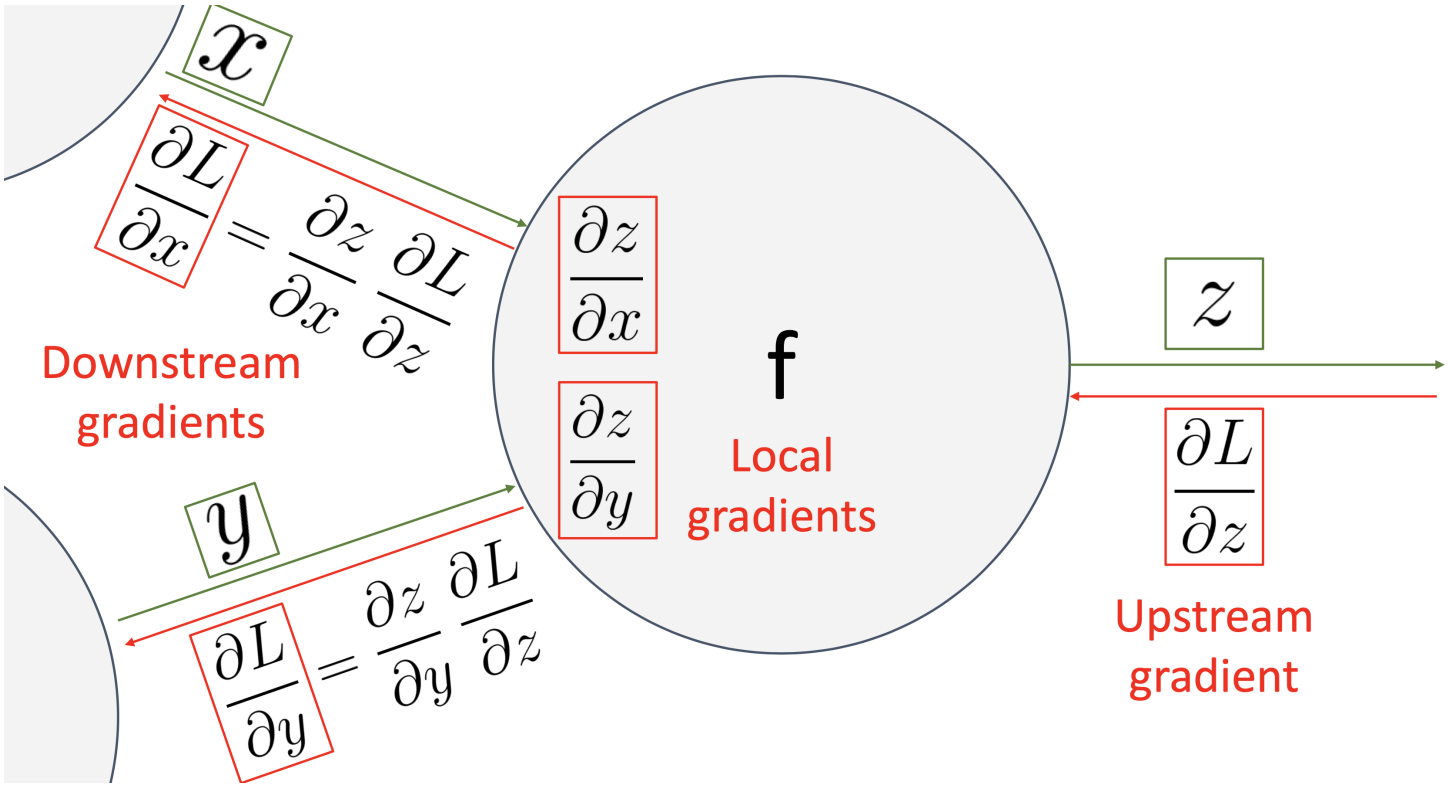
Downstream Gradient | **Local Gradient** | **Upstream Gradient**


x
f
z
y

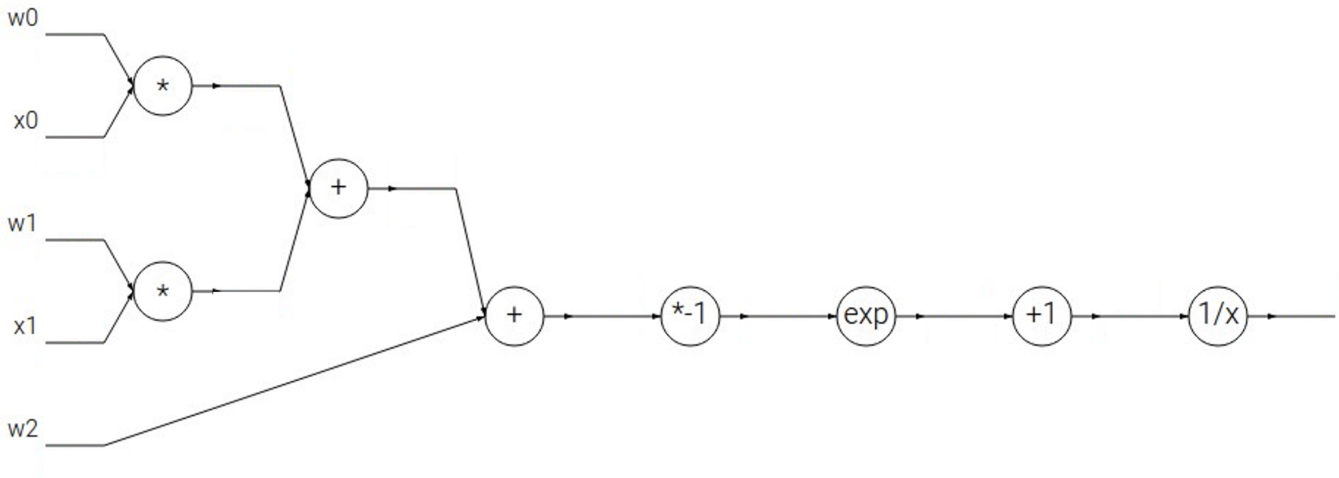

x
y
f
z
$\frac{\partial L}{\partial z}$
Upstream gradient


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

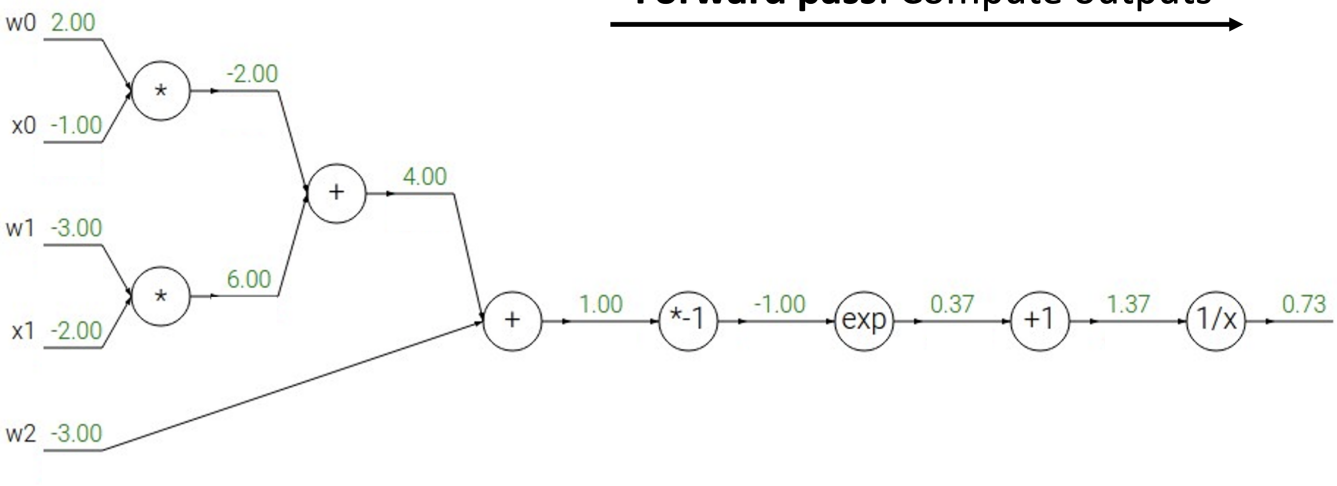

x
$\frac{\partial L}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial L}{\partial z}$
Downstream gradients
y
$\frac{\partial L}{\partial y} = \frac{\partial z}{\partial y}\frac{\partial L}{\partial z}$
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
Local gradients
z
$\frac{\partial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

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



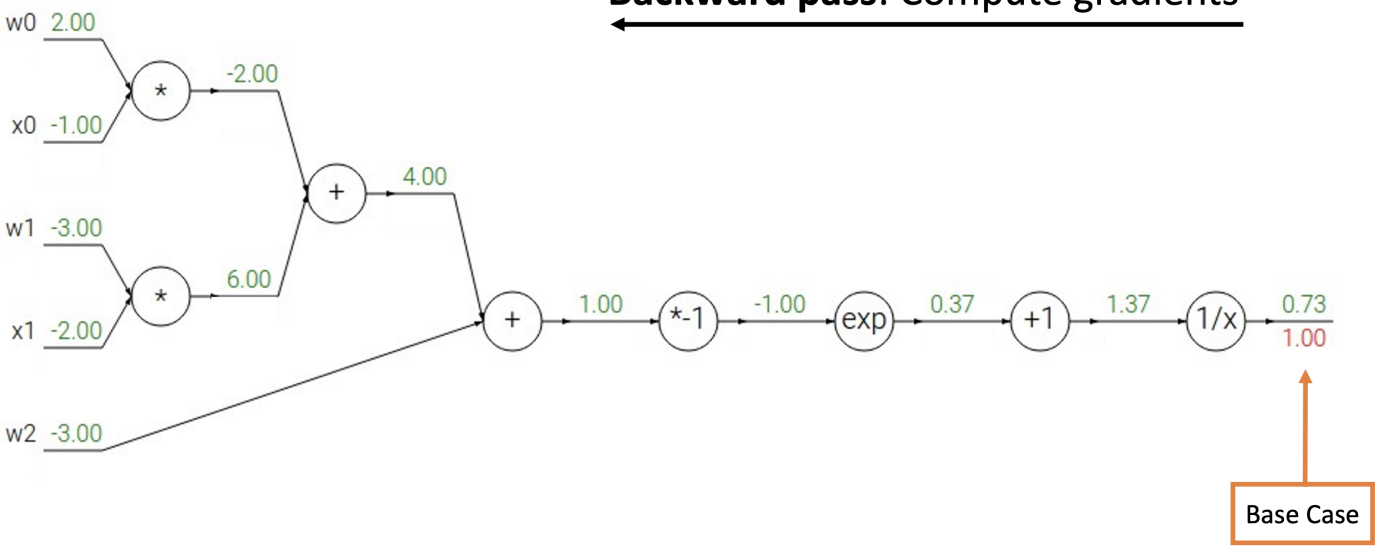
Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

Forward pass: Compute outputs



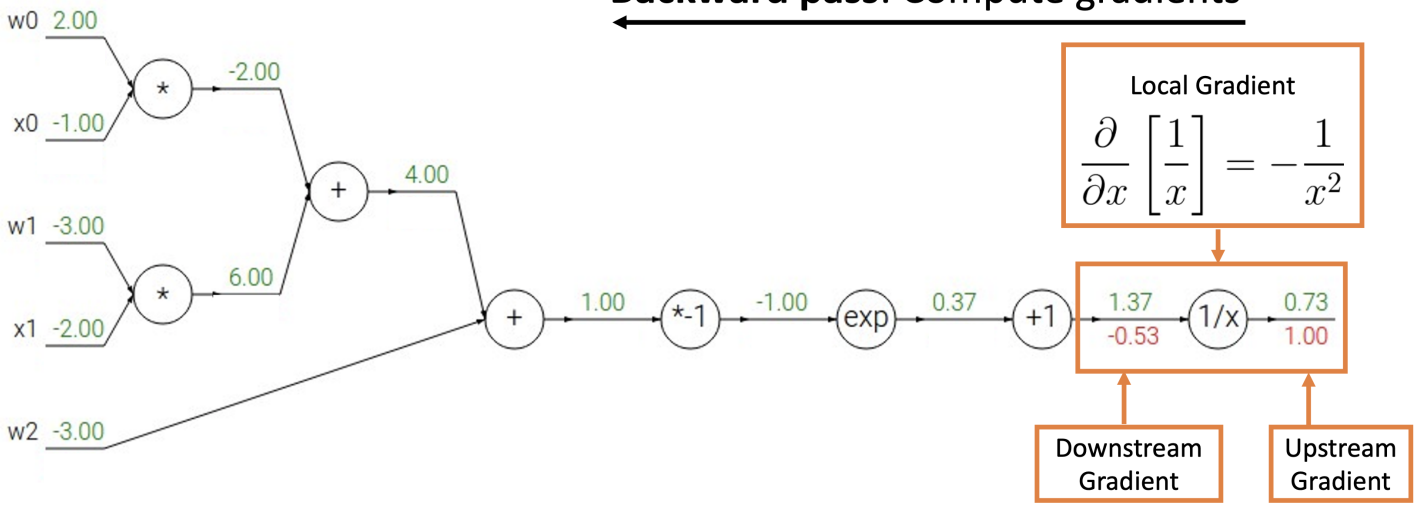
Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

Backward pass: Compute gradients



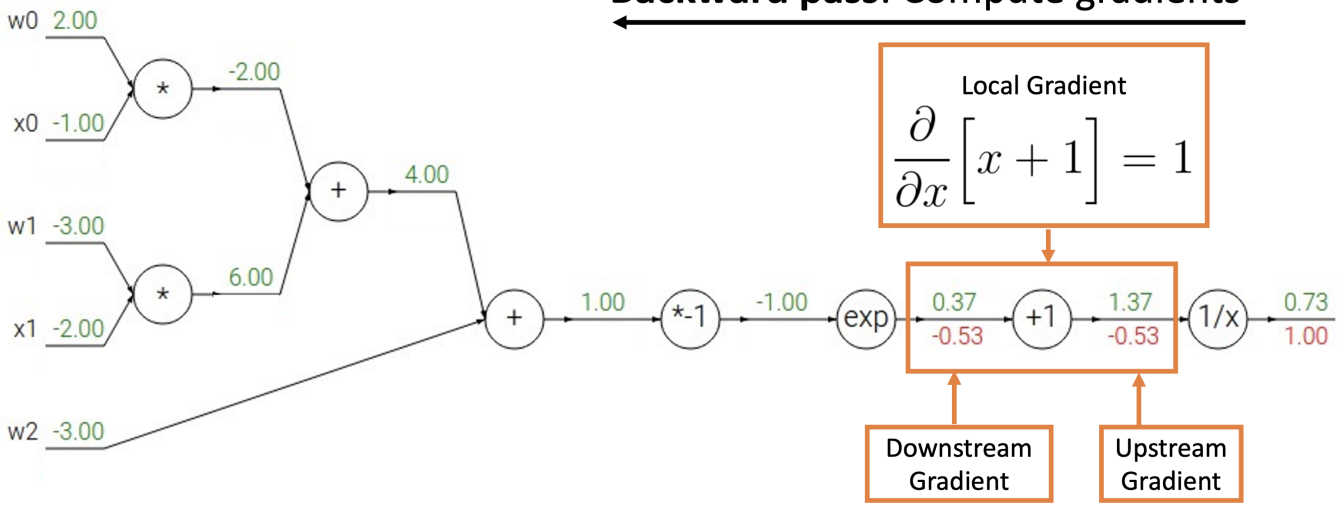
Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

Backward pass: Compute gradients

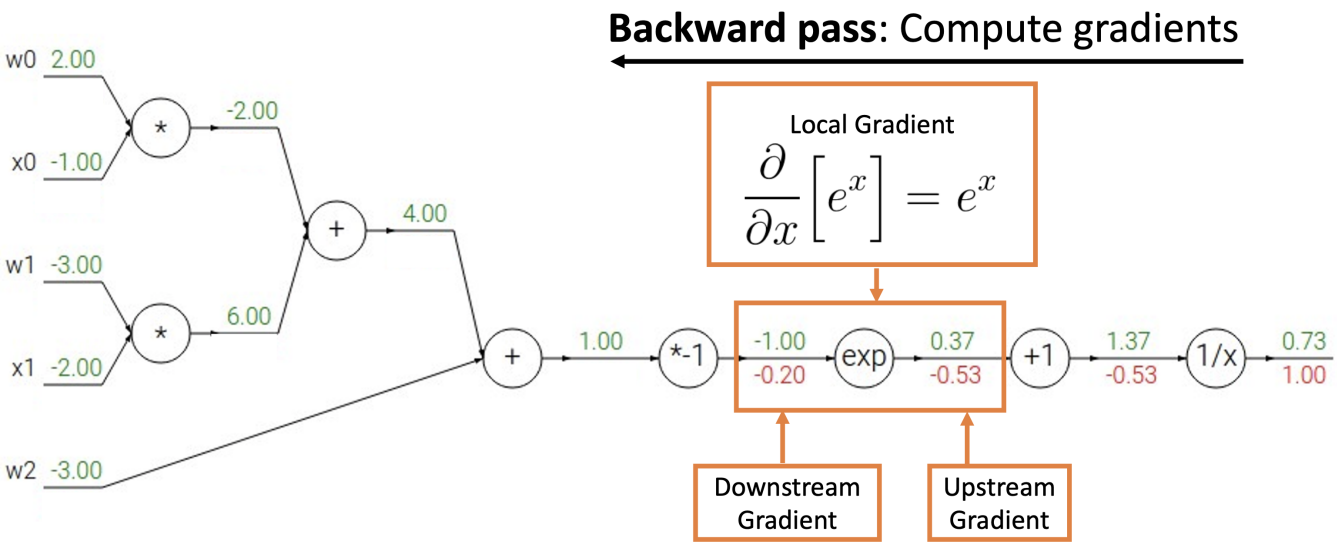


Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

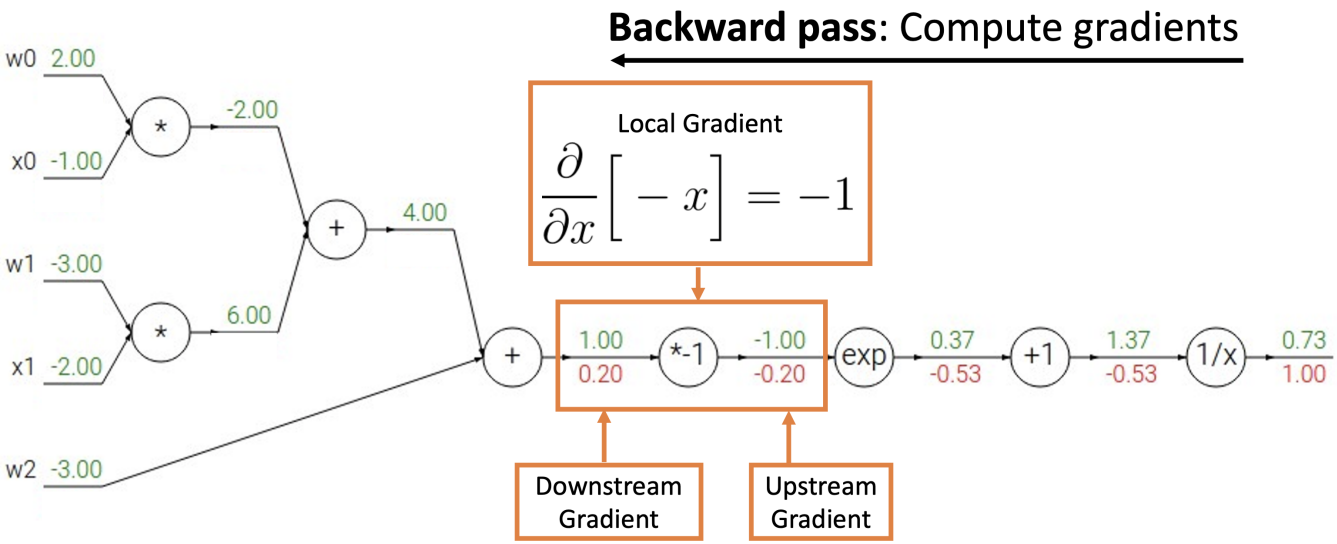
Backward pass: Compute gradients



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

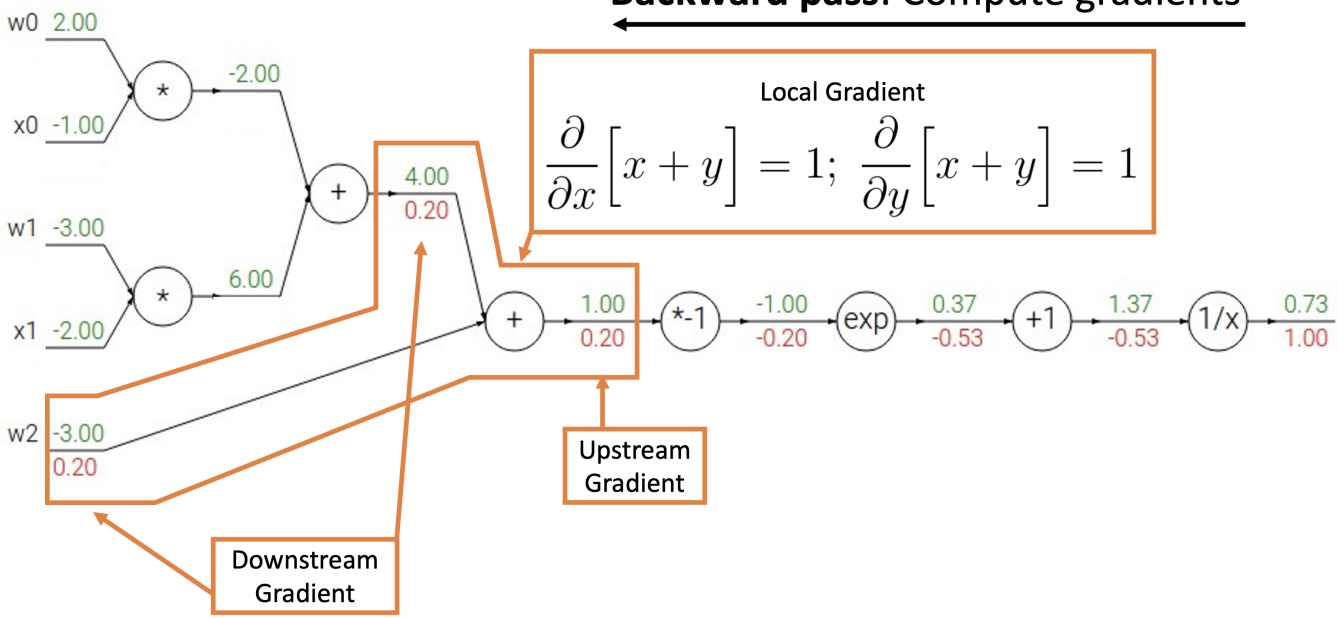


Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



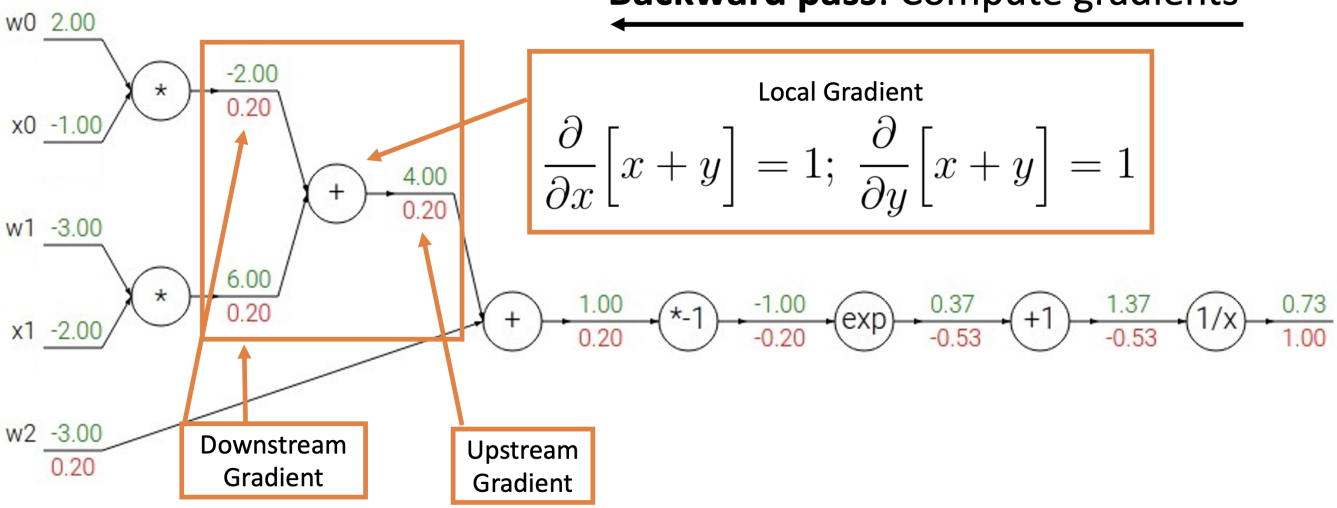
Another Example $f(x,w) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

Backward pass: Compute gradients



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

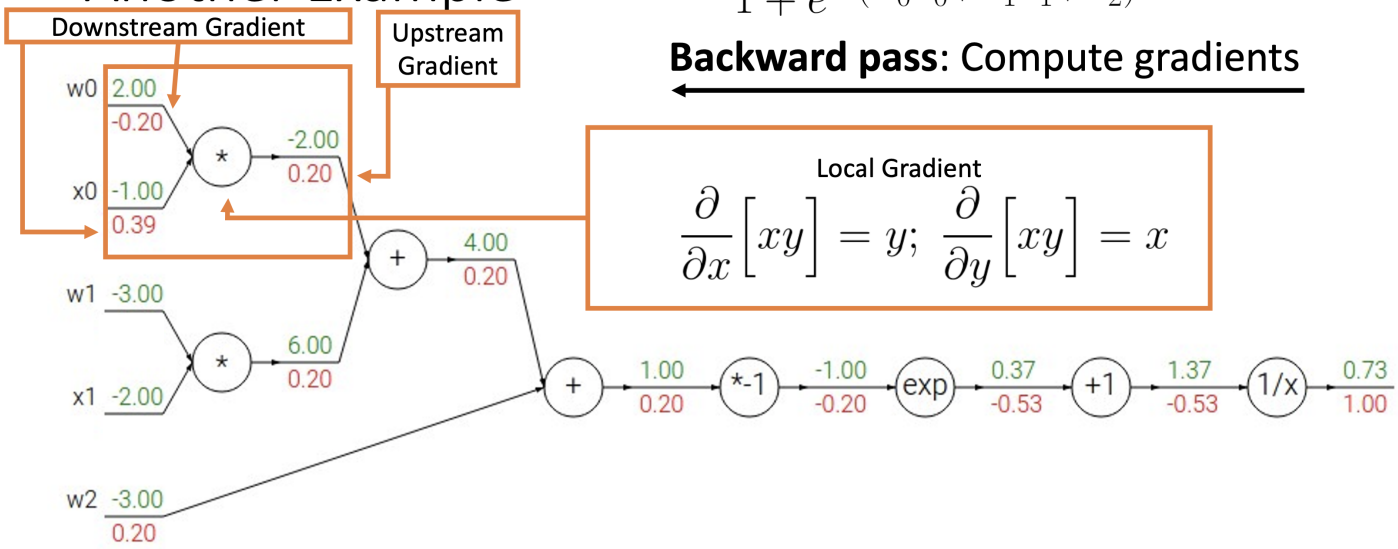
Backward pass: Compute gradients



Another Example

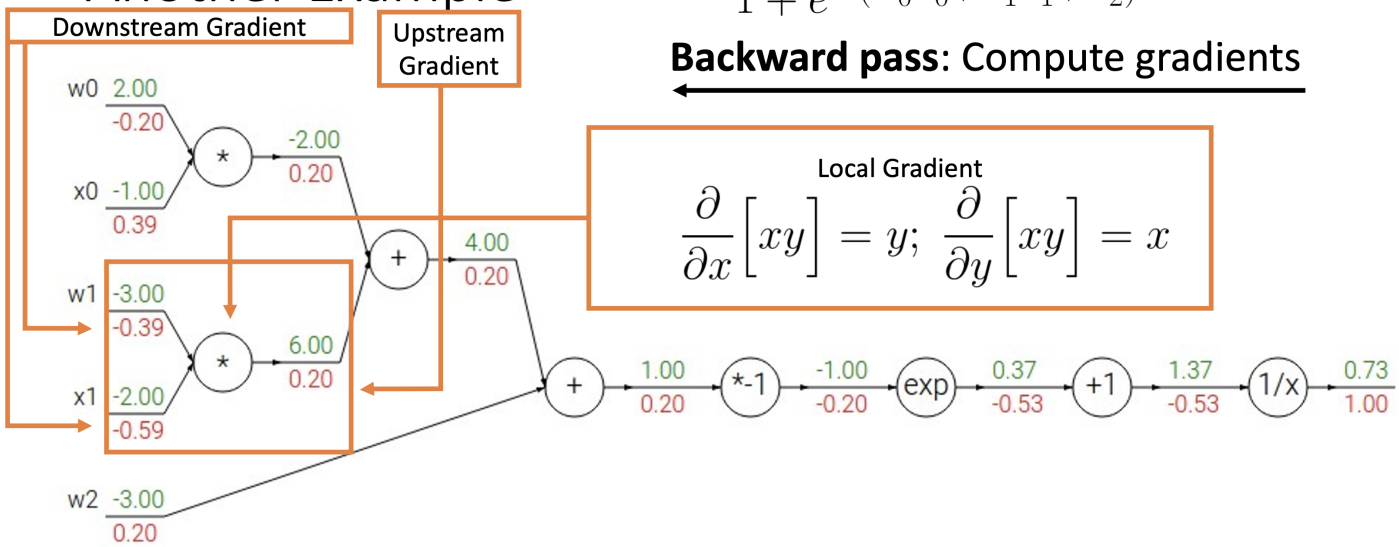
$$f(x,w) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

Backward pass: Compute gradients



Another Example

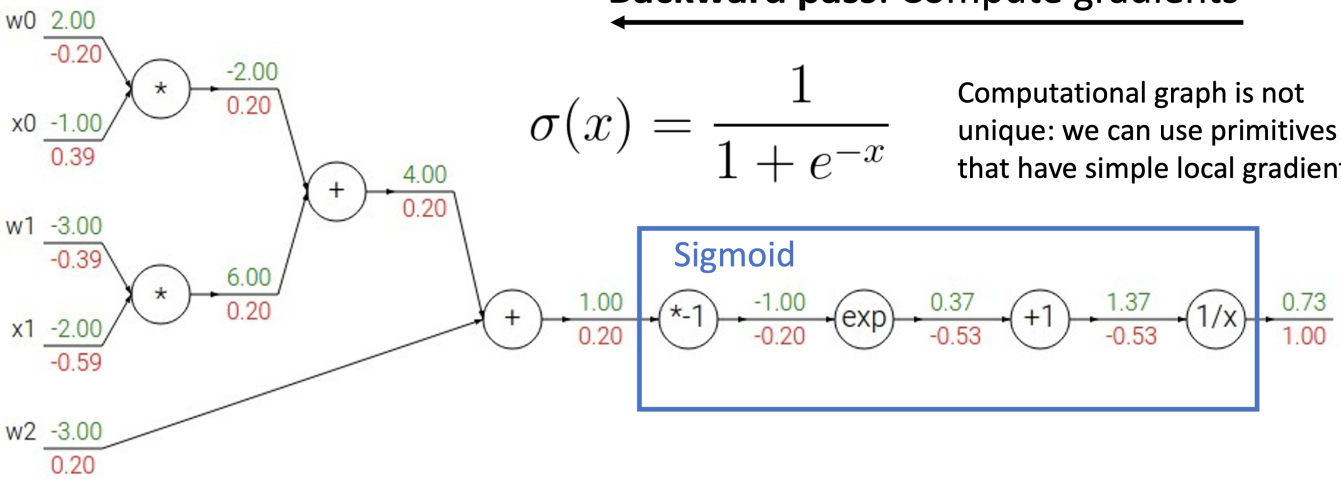
$$f(x,w) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



Another Example

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

Backward pass: Compute gradients



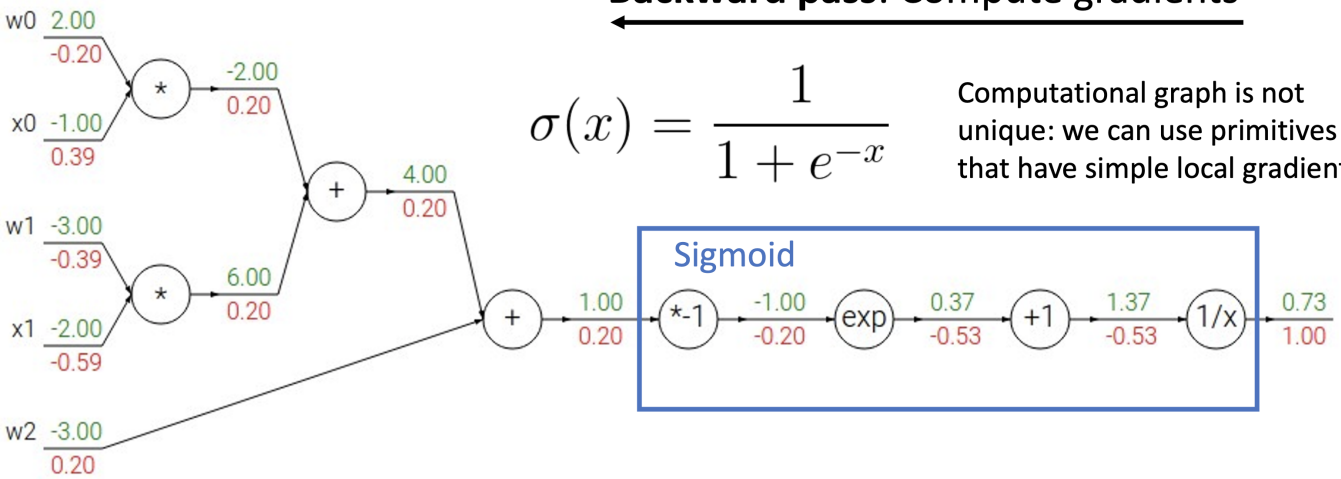
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

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

Another Example

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

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

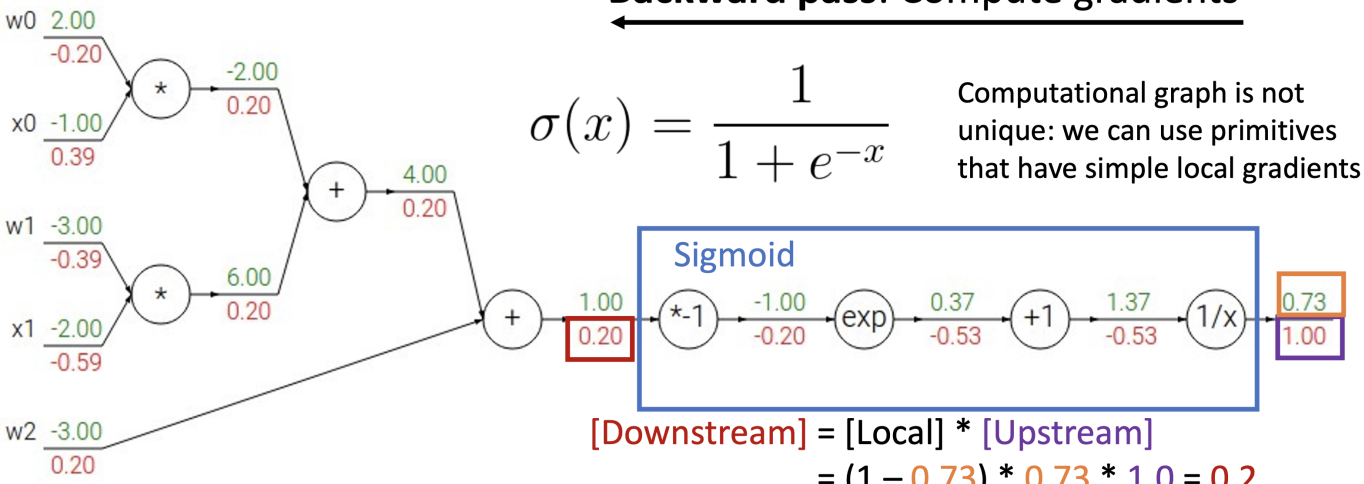
Sigmoid local gradient:

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

Another Example

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

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:

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

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?

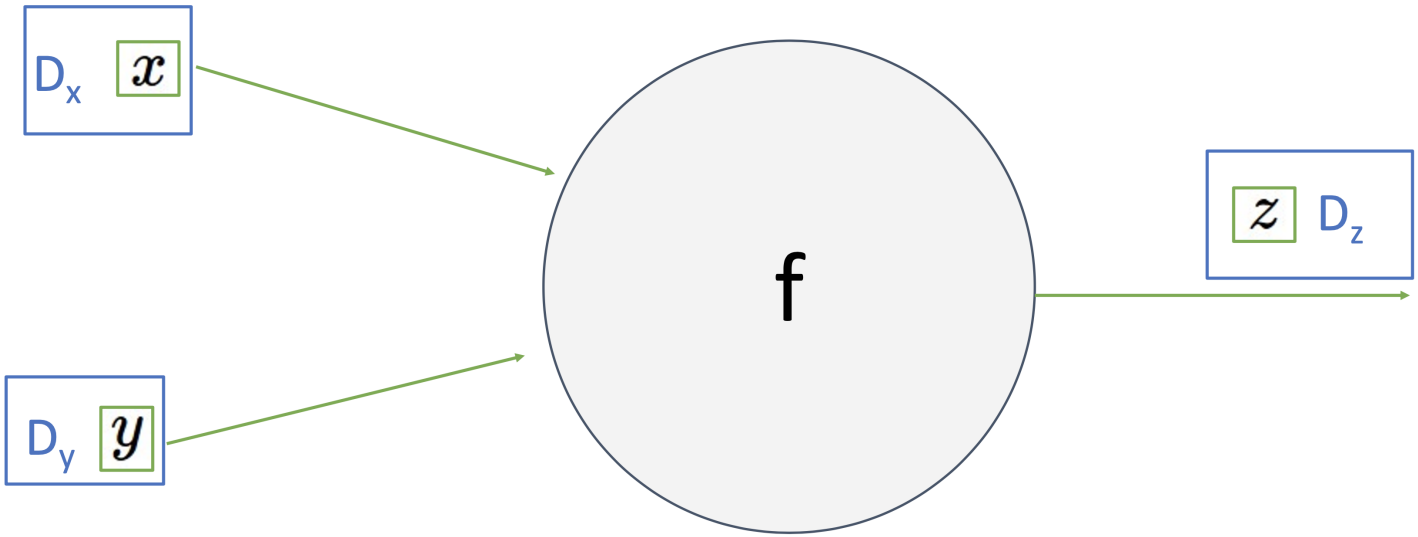
$x \in \mathbb{R}^N, y \in \mathbb{R}^M$

Derivative is **Jacobian**:

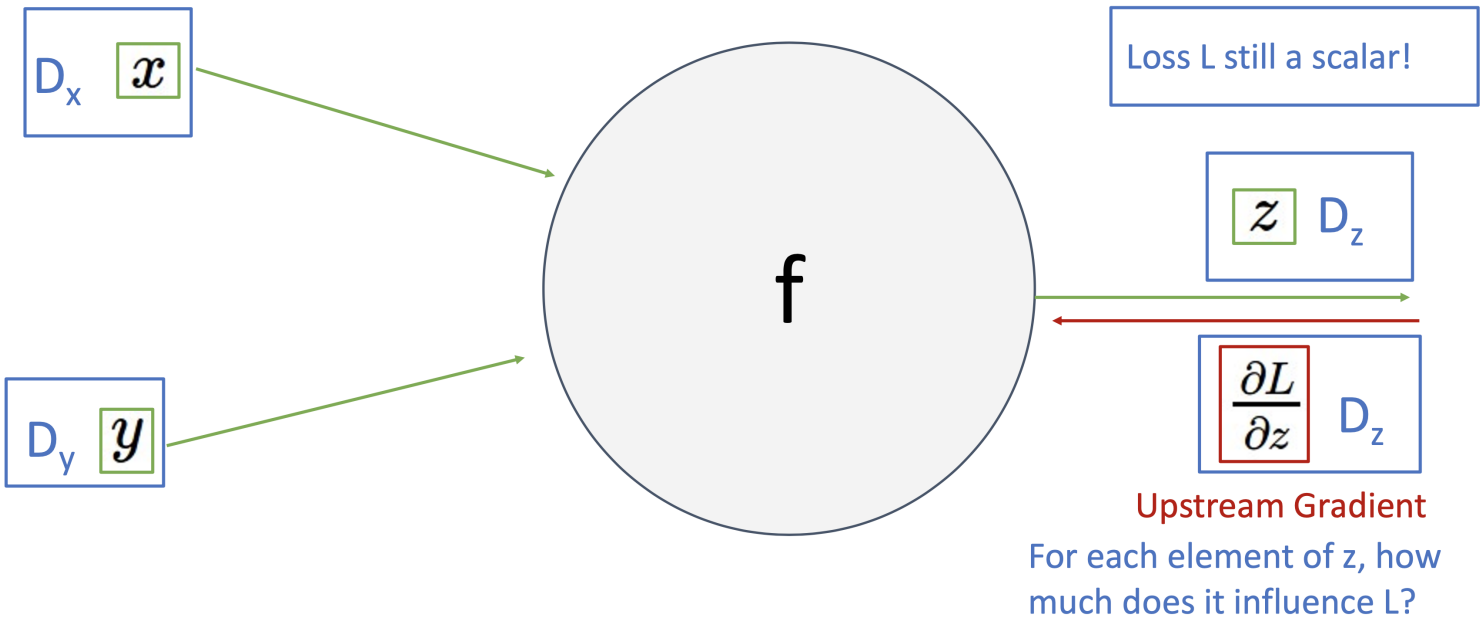
$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

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

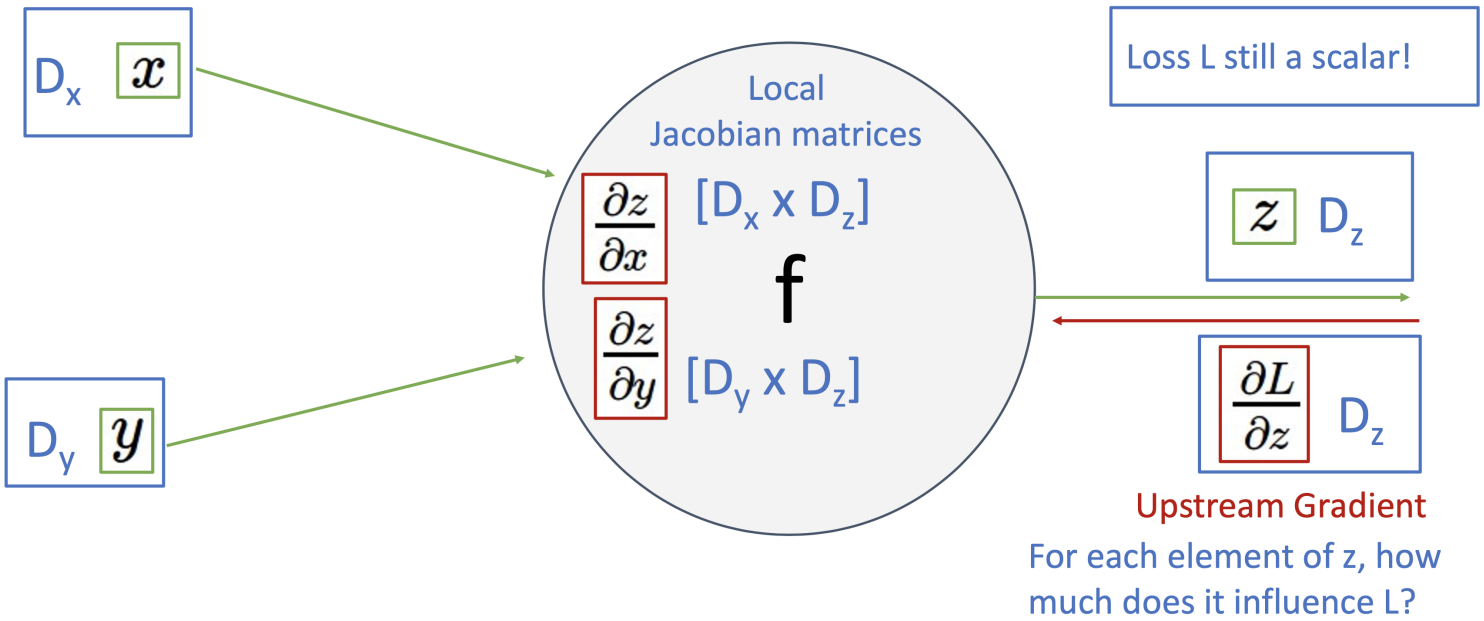
Backprop with Vectors



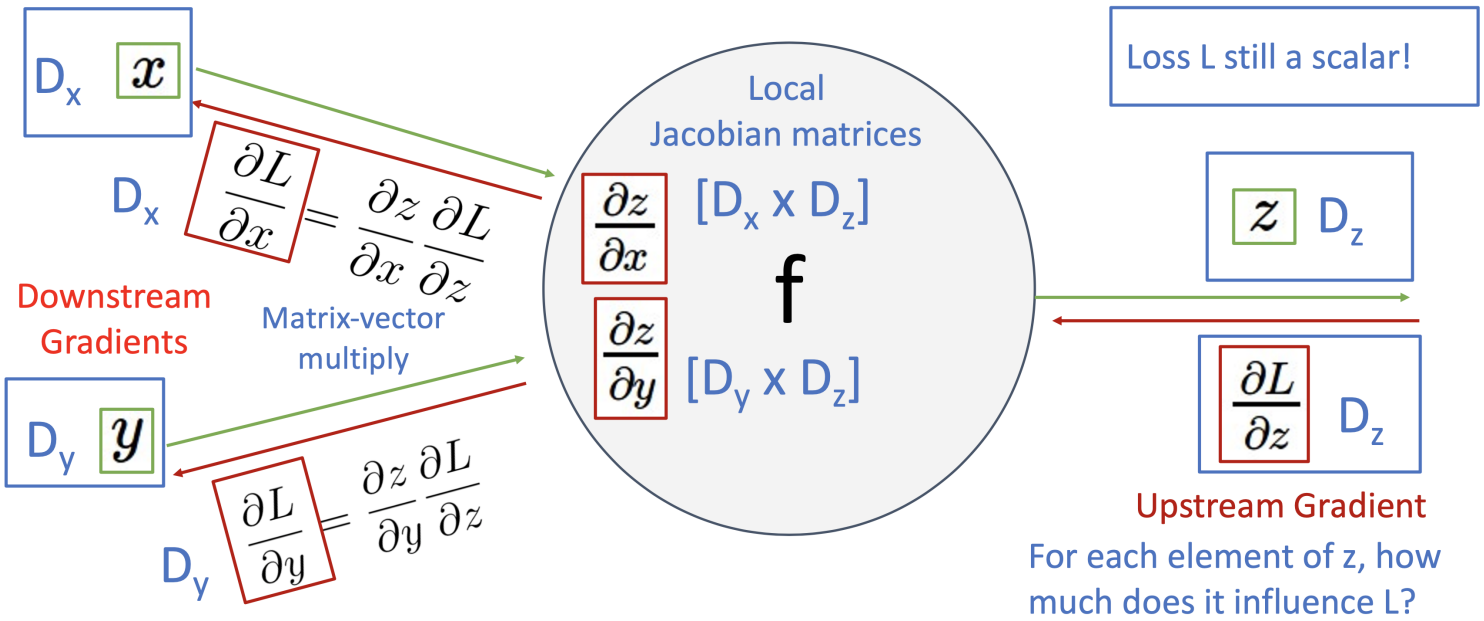
Backprop with Vectors



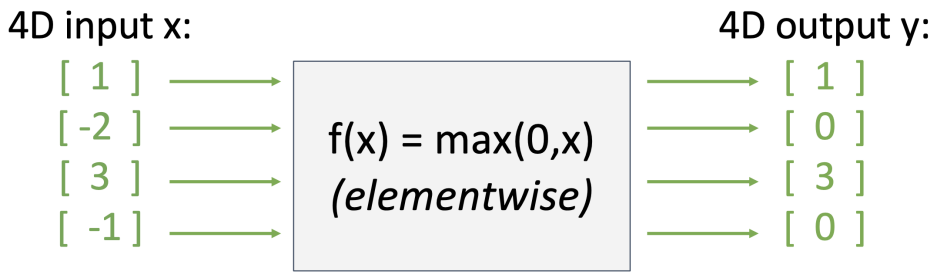
Backprop with Vectors



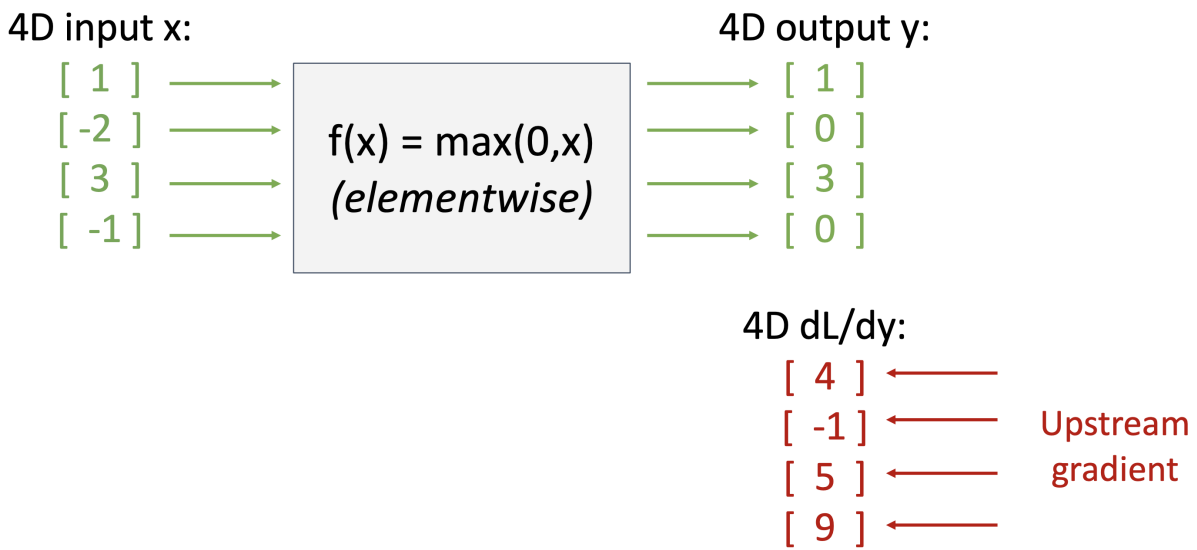
Backprop with Vectors



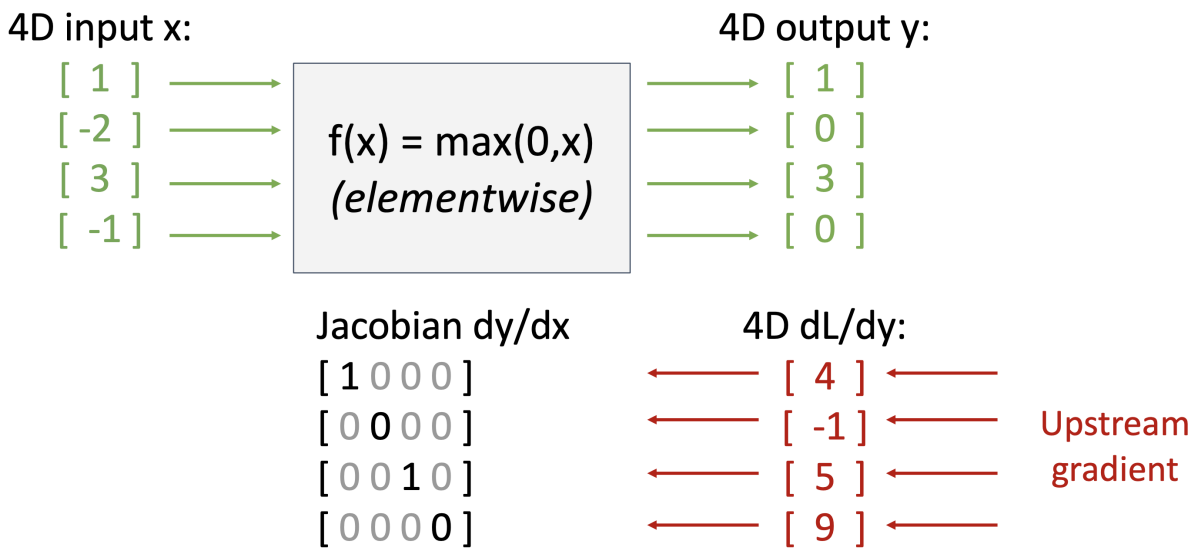
Backprop with Vectors



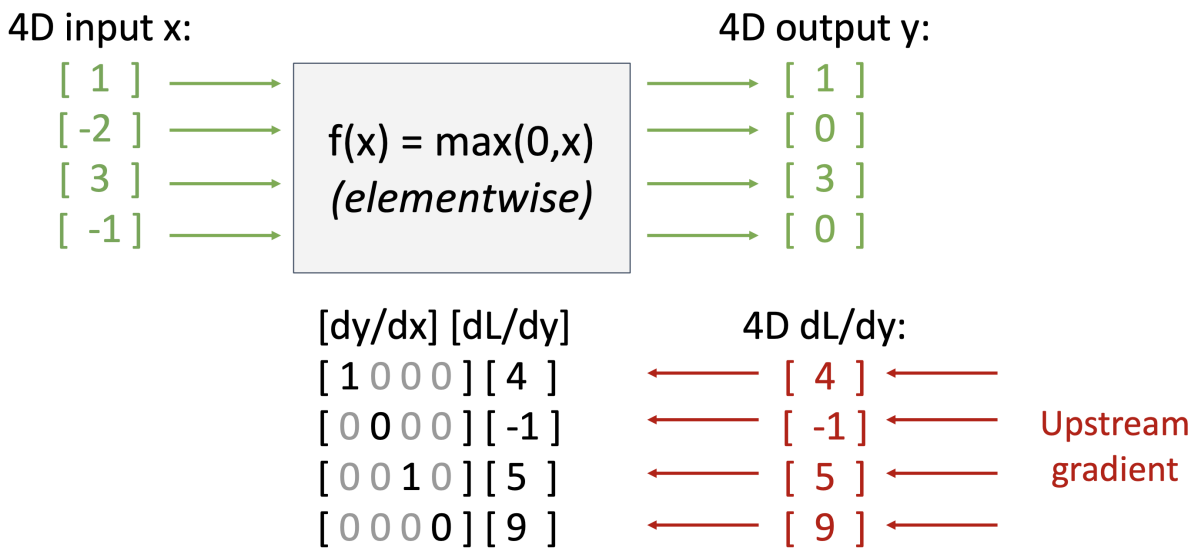
Backprop with Vectors



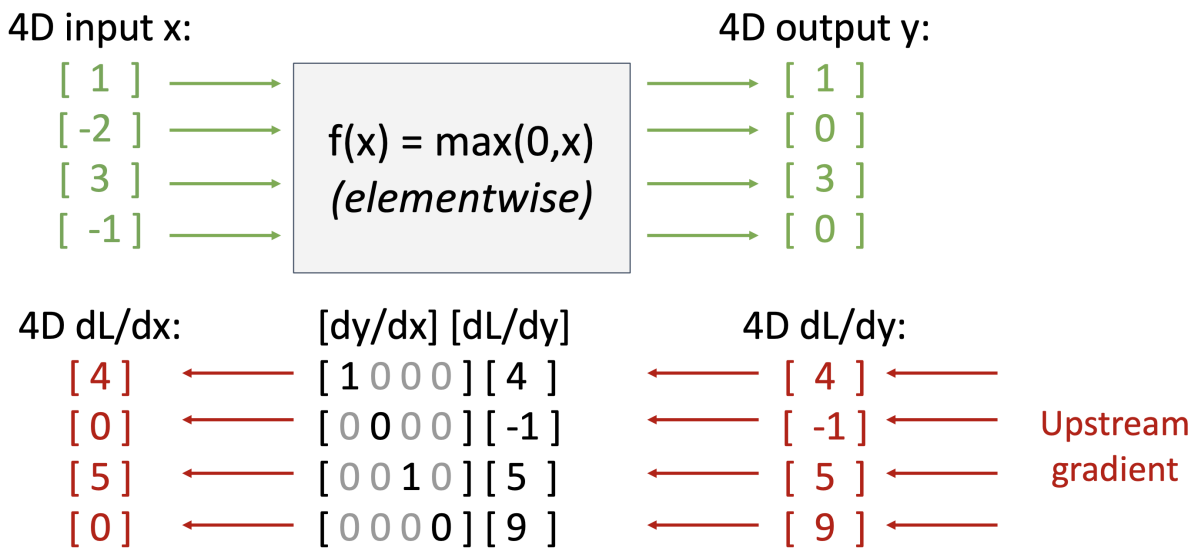
Backprop with Vectors



Backprop with Vectors

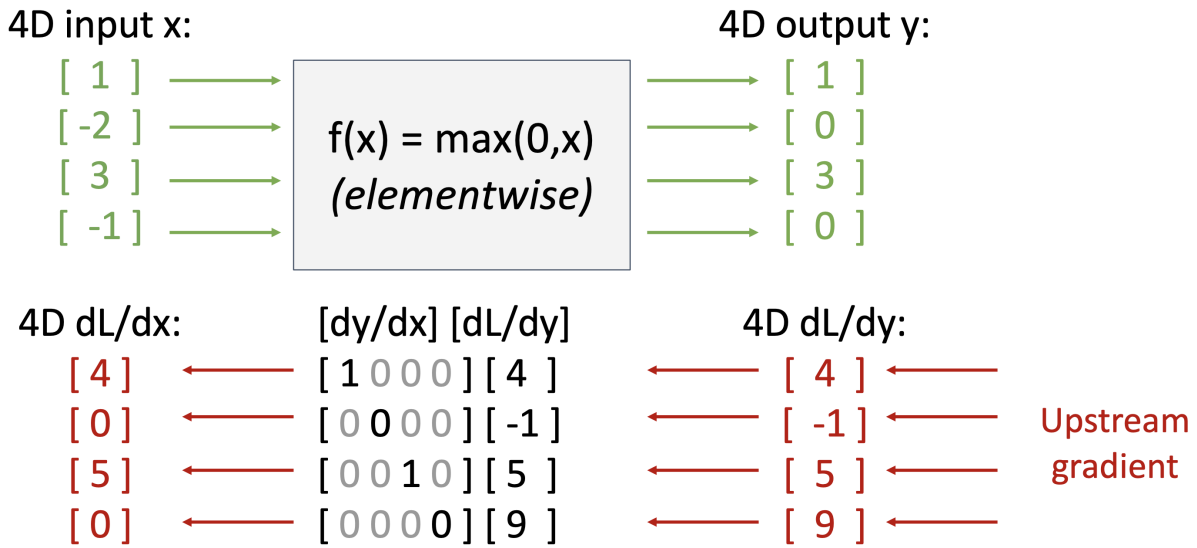


Backprop with Vectors



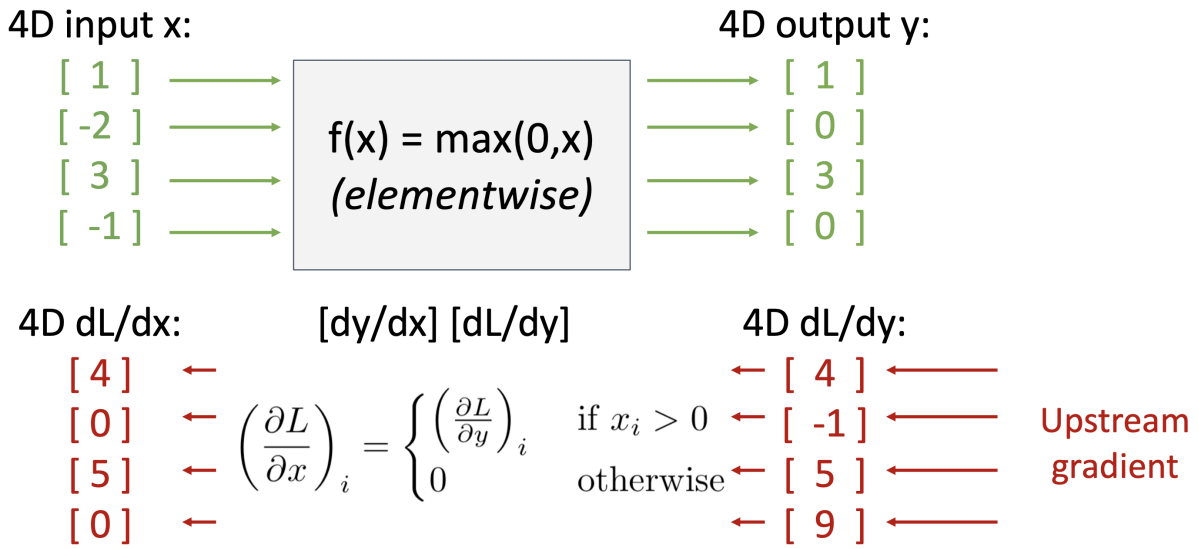
Backprop with Vectors

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

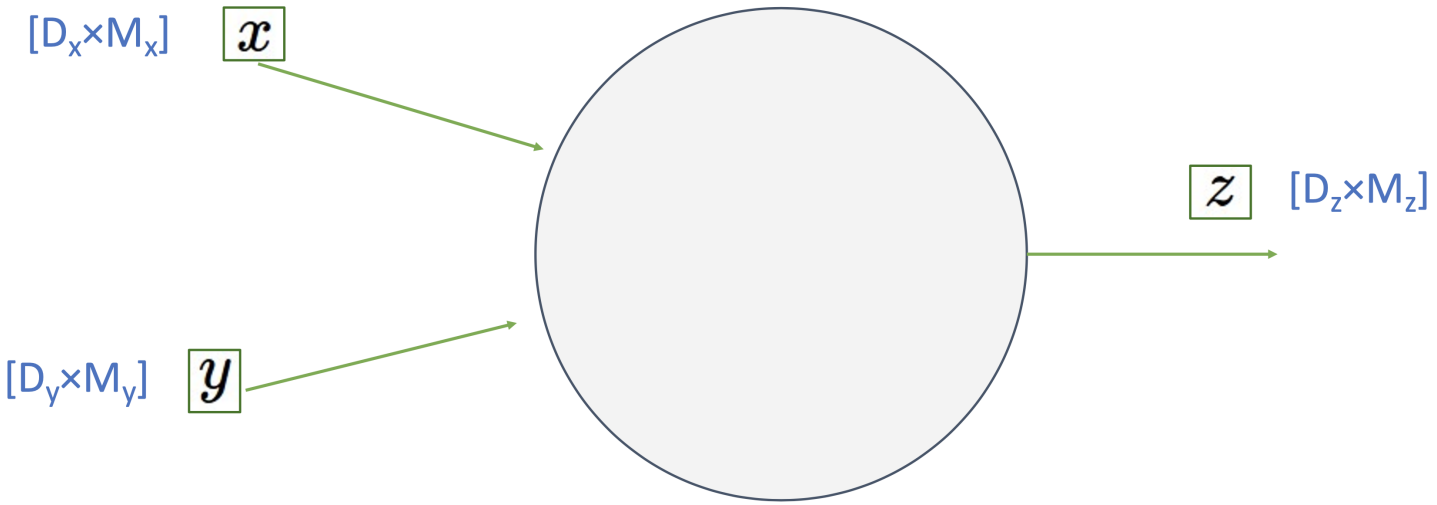


Backprop with Vectors

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



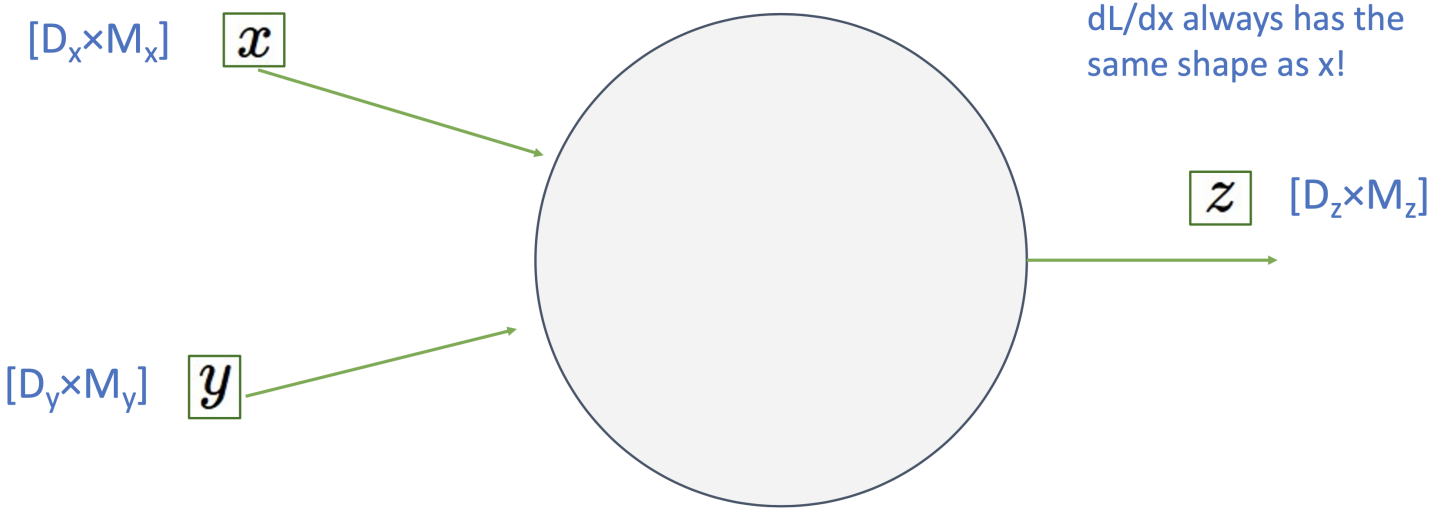
Backprop with Matrices (or Tensors):



Backprop with Matrices (or Tensors):

Loss L still a scalar!

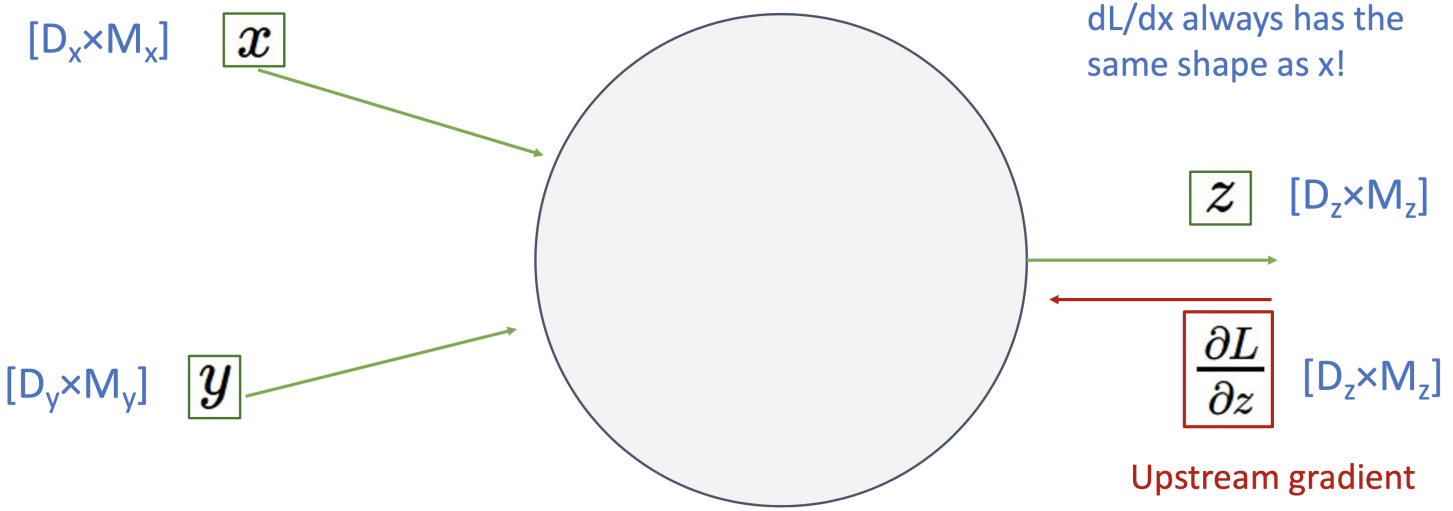
dL/dx always has the same shape as x!



Backprop with Matrices (or Tensors):

Loss L still a scalar!

dL/dx always has the same shape as x!



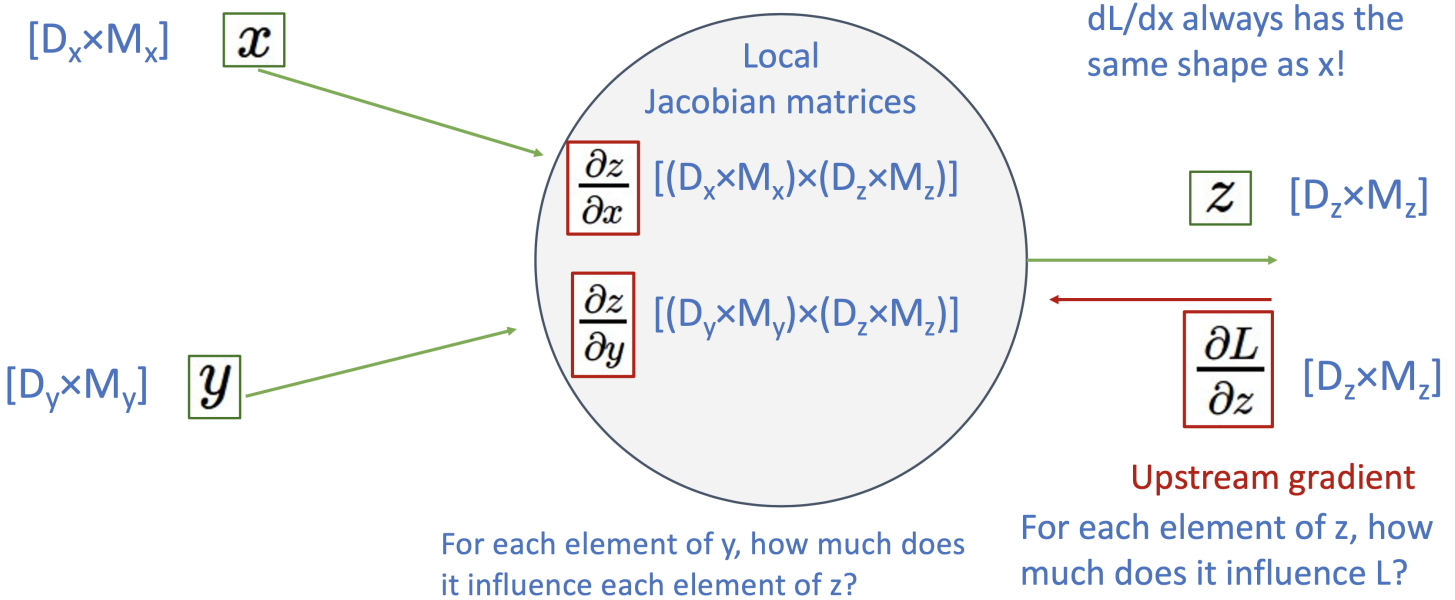
Upstream gradient

For each element of z, how much does it influence L?

Backprop with Matrices (or Tensors):

Loss L still a scalar!

dL/dx always has the same shape as x!



For each element of y, how much does it influence each element of z?

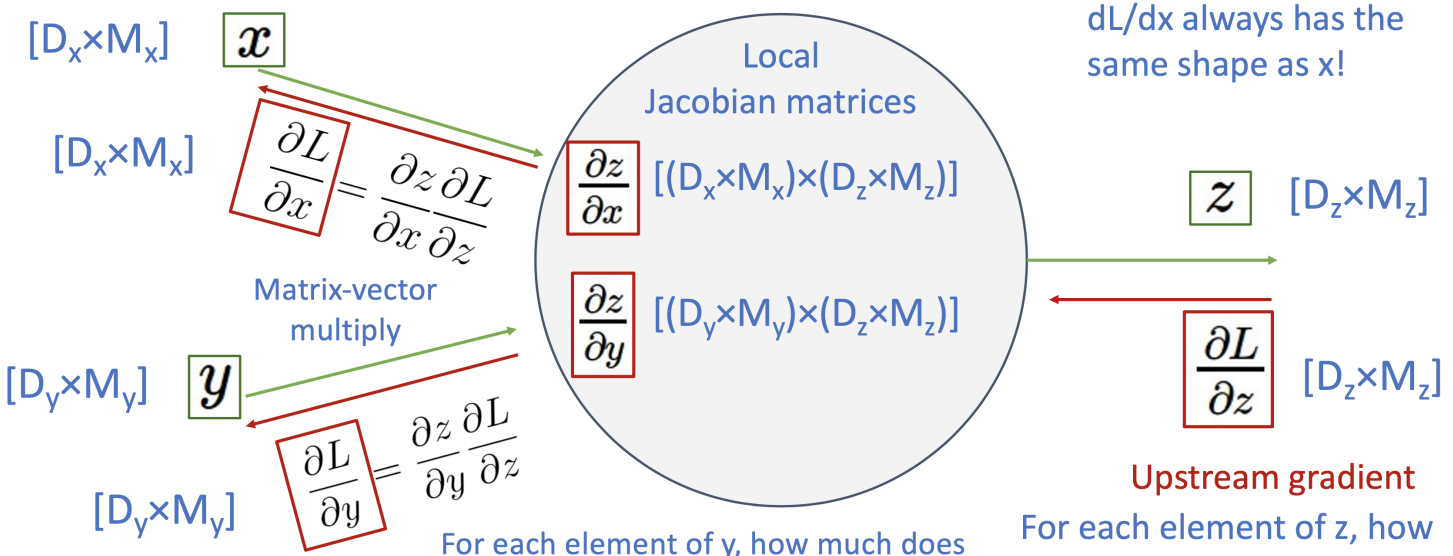
Upstream gradient

For each element of z, how much does it influence L?

Backprop with Matrices (or Tensors):

Loss L still a scalar!

dL/dx always has the same shape as x!



For each element of y, how much does it influence each element of z?

Upstream gradient

For each element of z, how much does it influence L?

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]

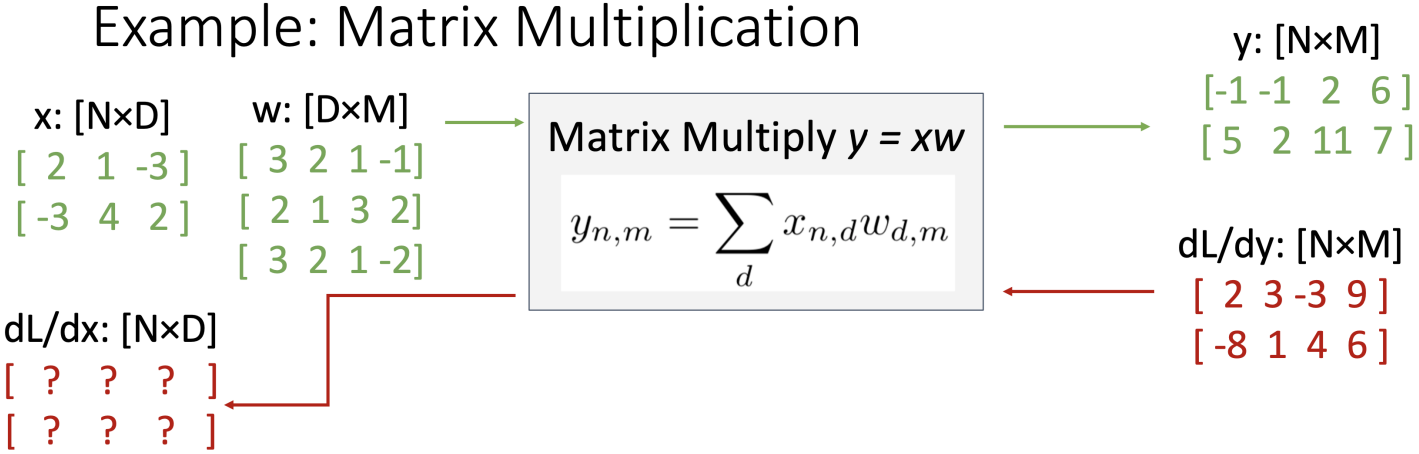
Matrix Multiply *y = xw*

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

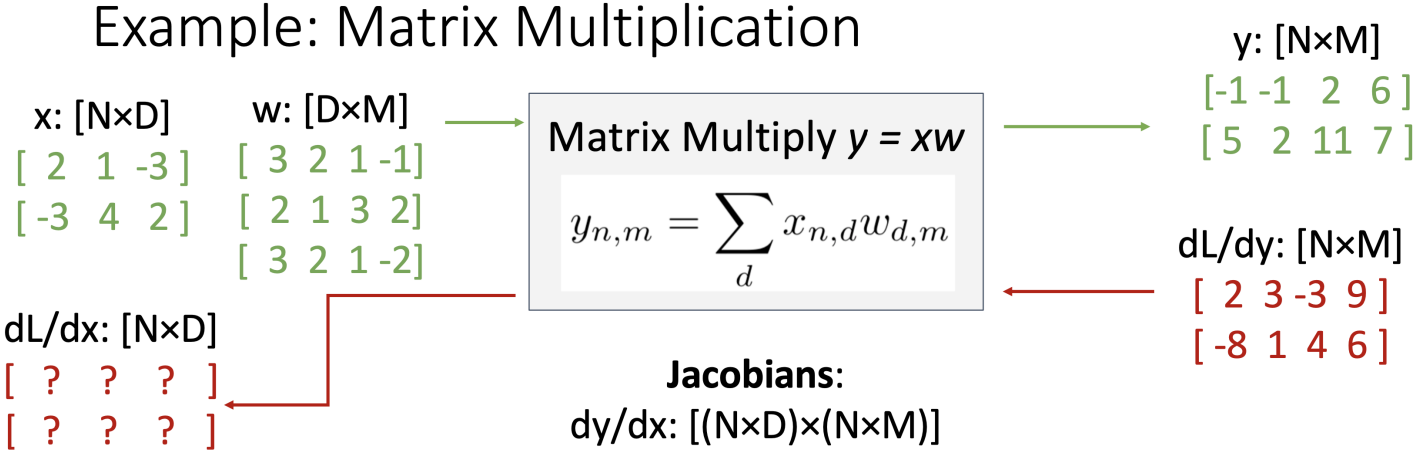
y: [N×M]

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

Example: Matrix Multiplication



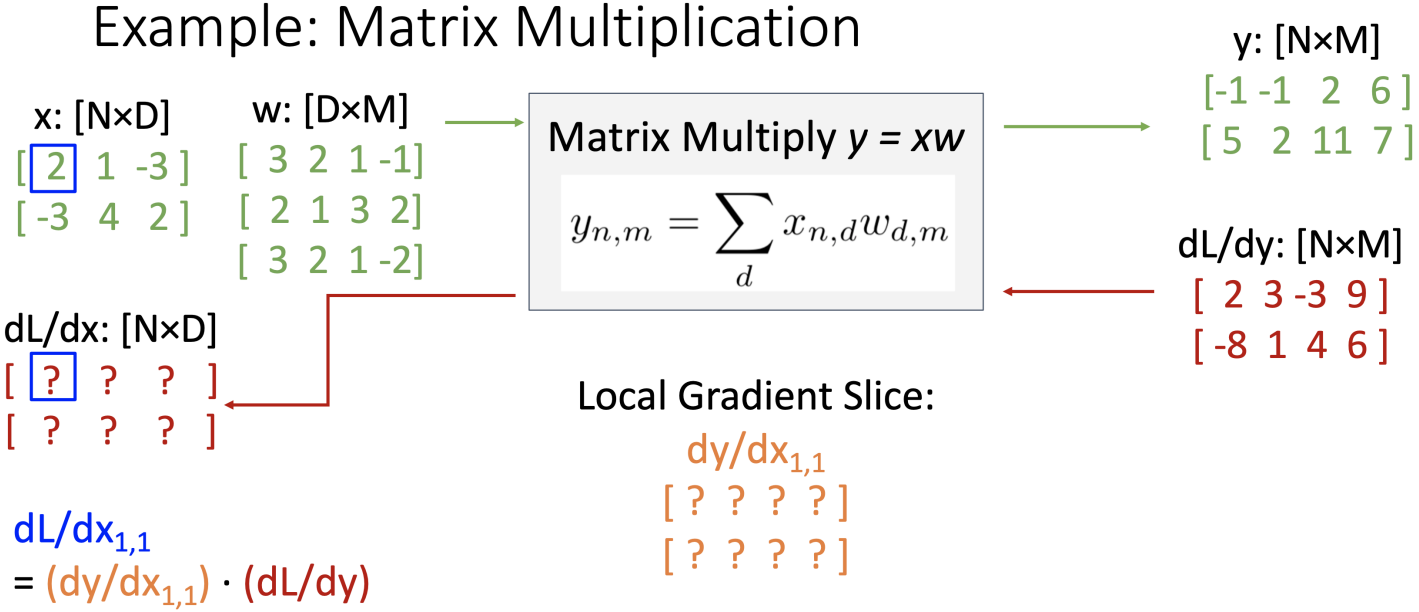
Example: Matrix Multiplication



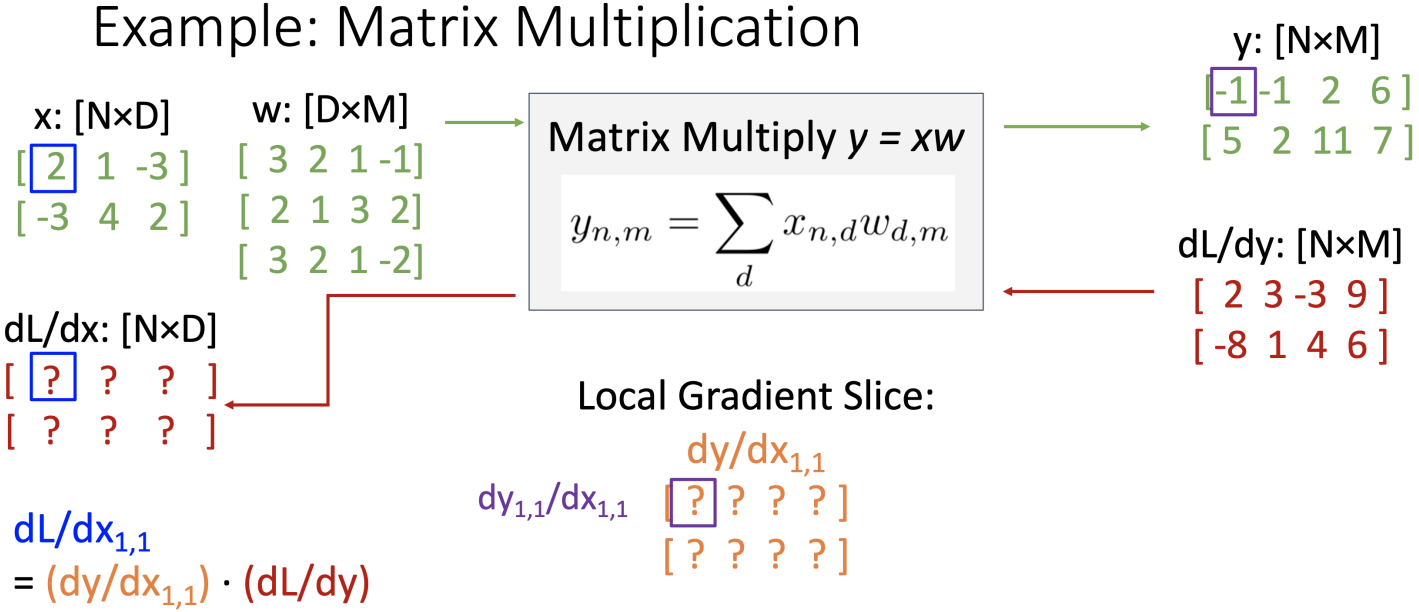
Jacobians:
dy/dx: [(N×D)×(N×M)]
dy/dw: [(D×M)×(N×M)]

For a neural net we may have
N=64, D=M=4096
Each Jacobian takes 256 GB of memory! Must work with them implicitly!

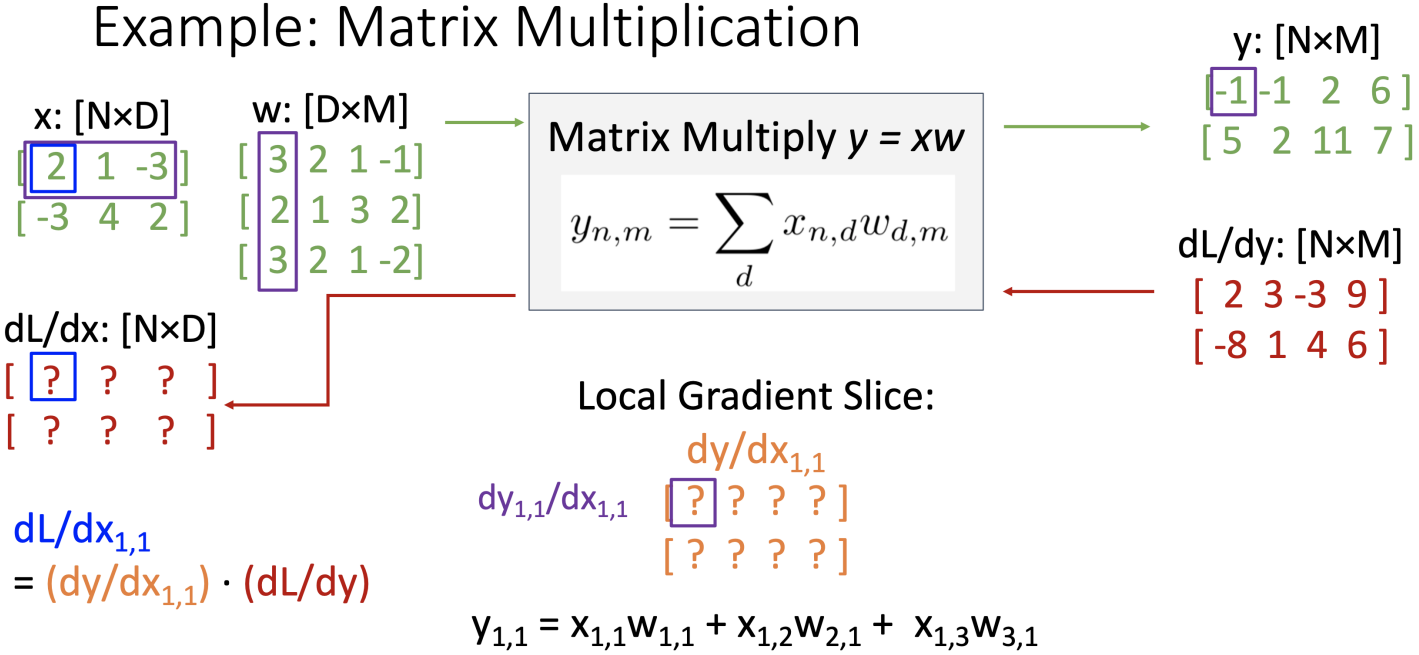
Example: Matrix Multiplication



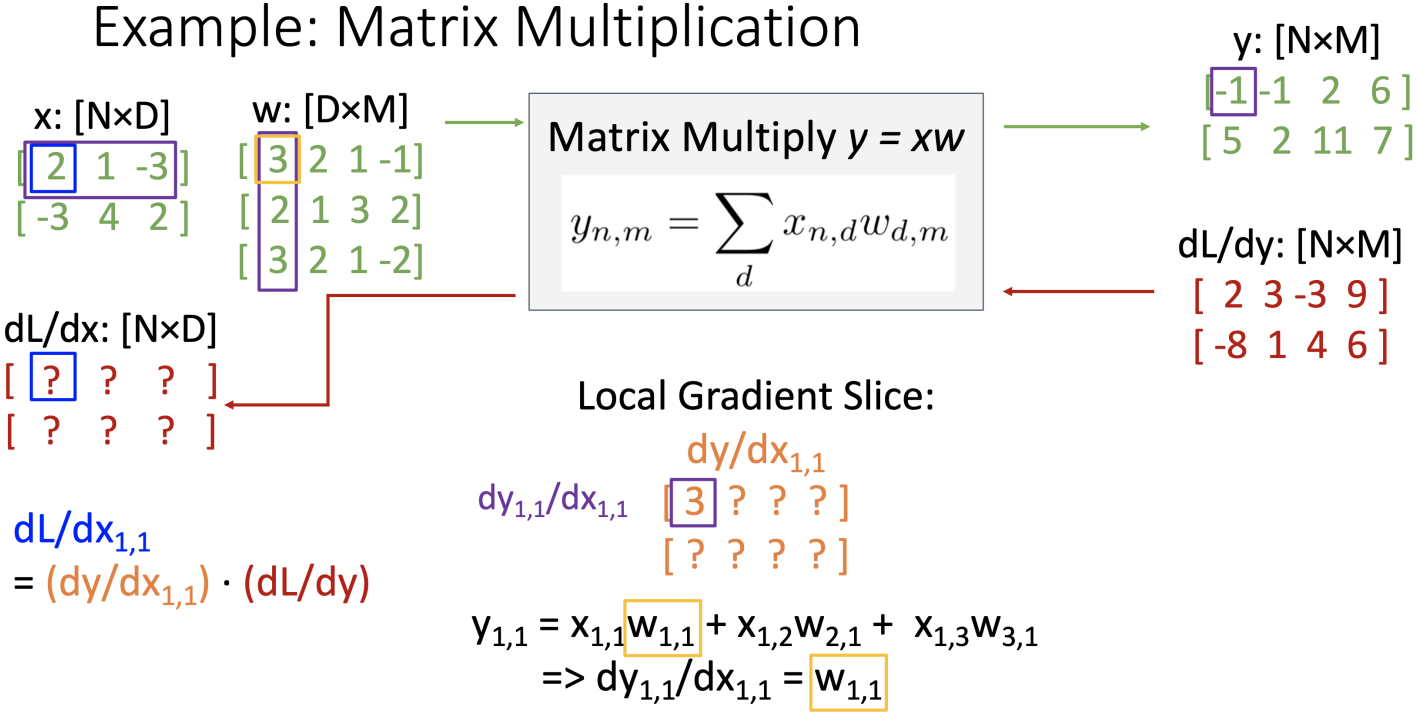
Example: Matrix Multiplication



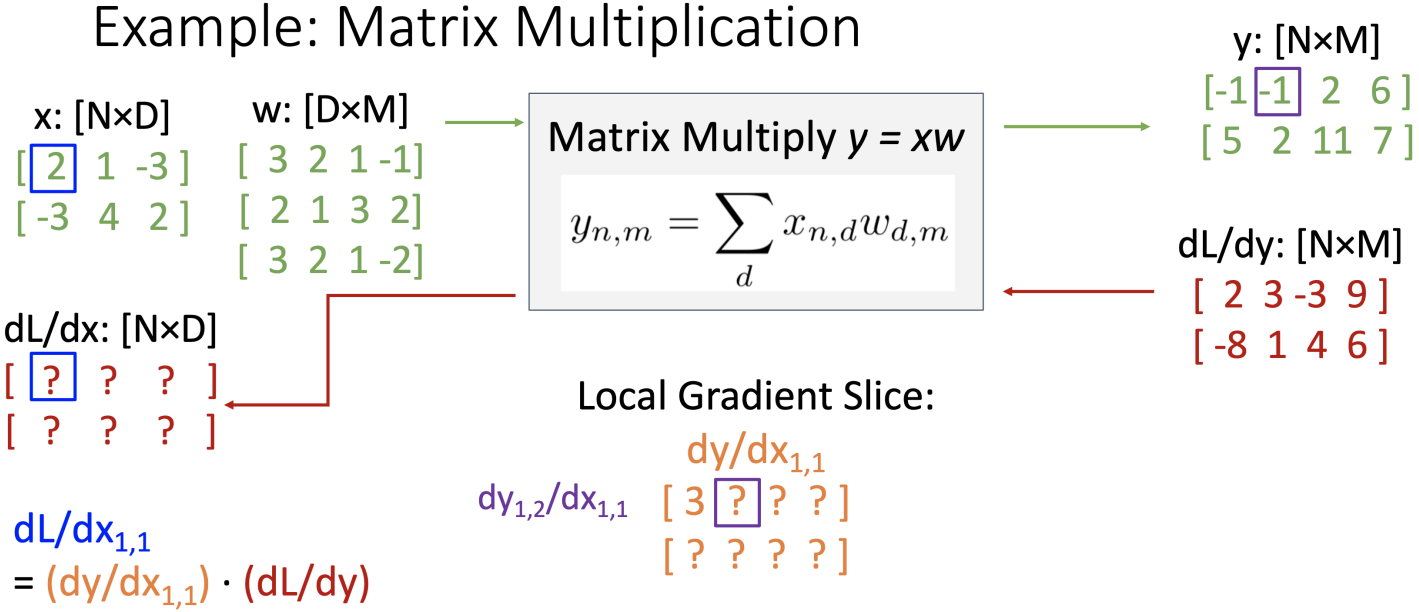
Example: Matrix Multiplication



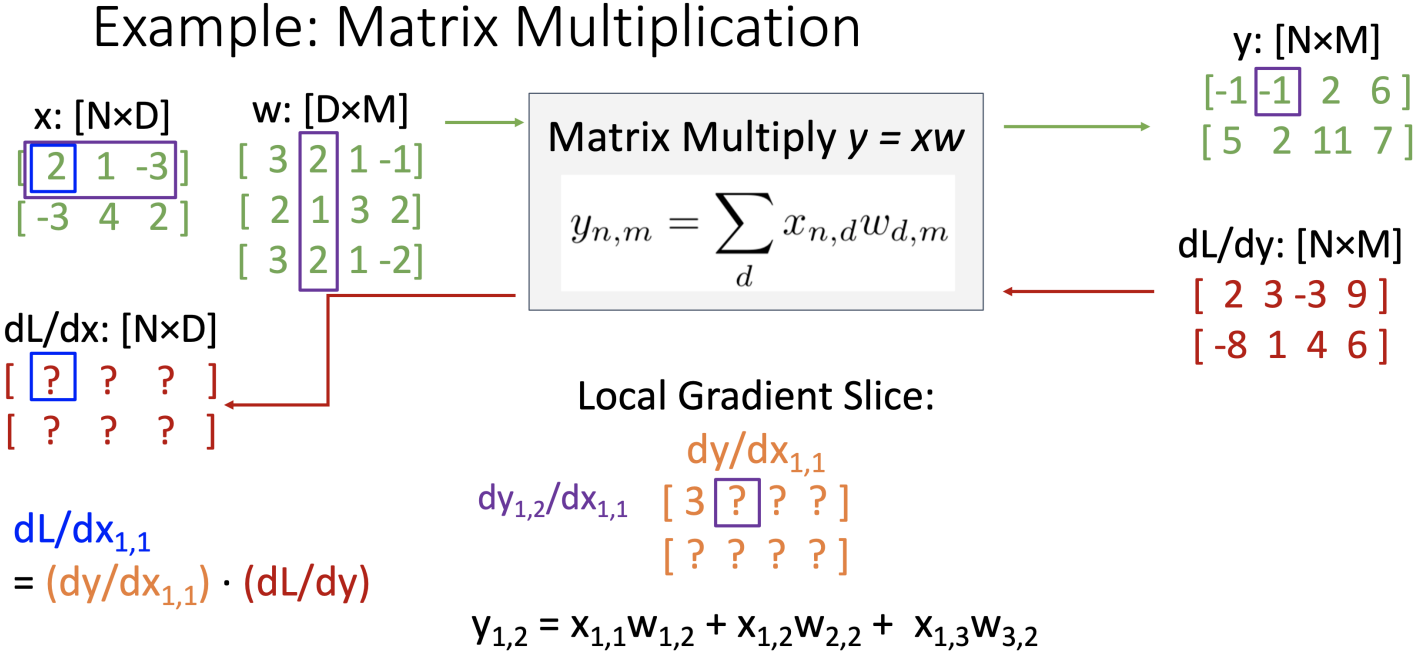
Example: Matrix Multiplication



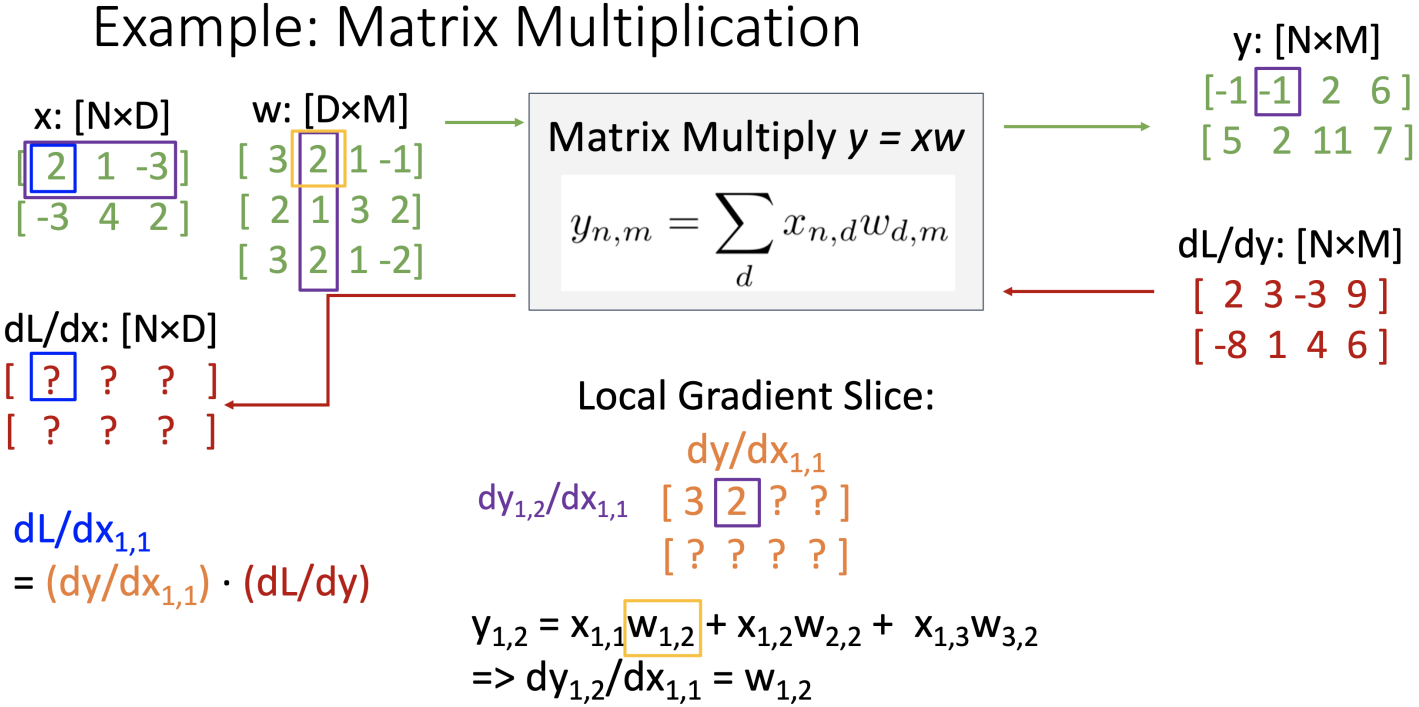
Example: Matrix Multiplication



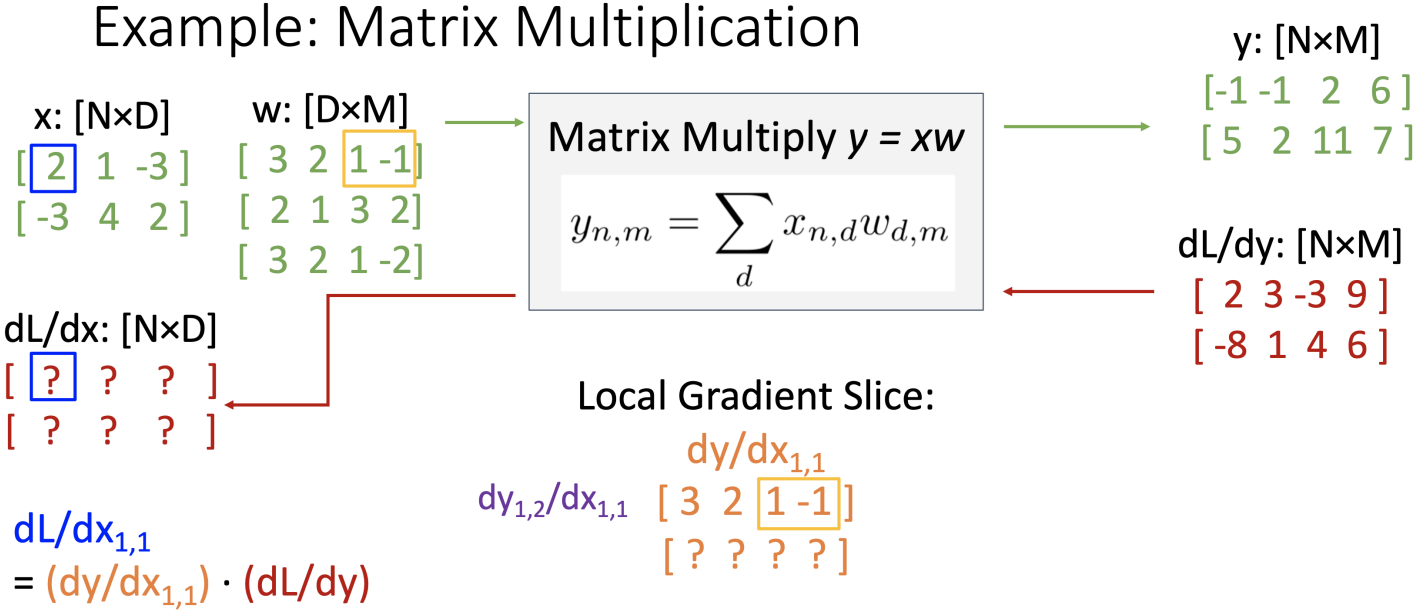
Example: Matrix Multiplication



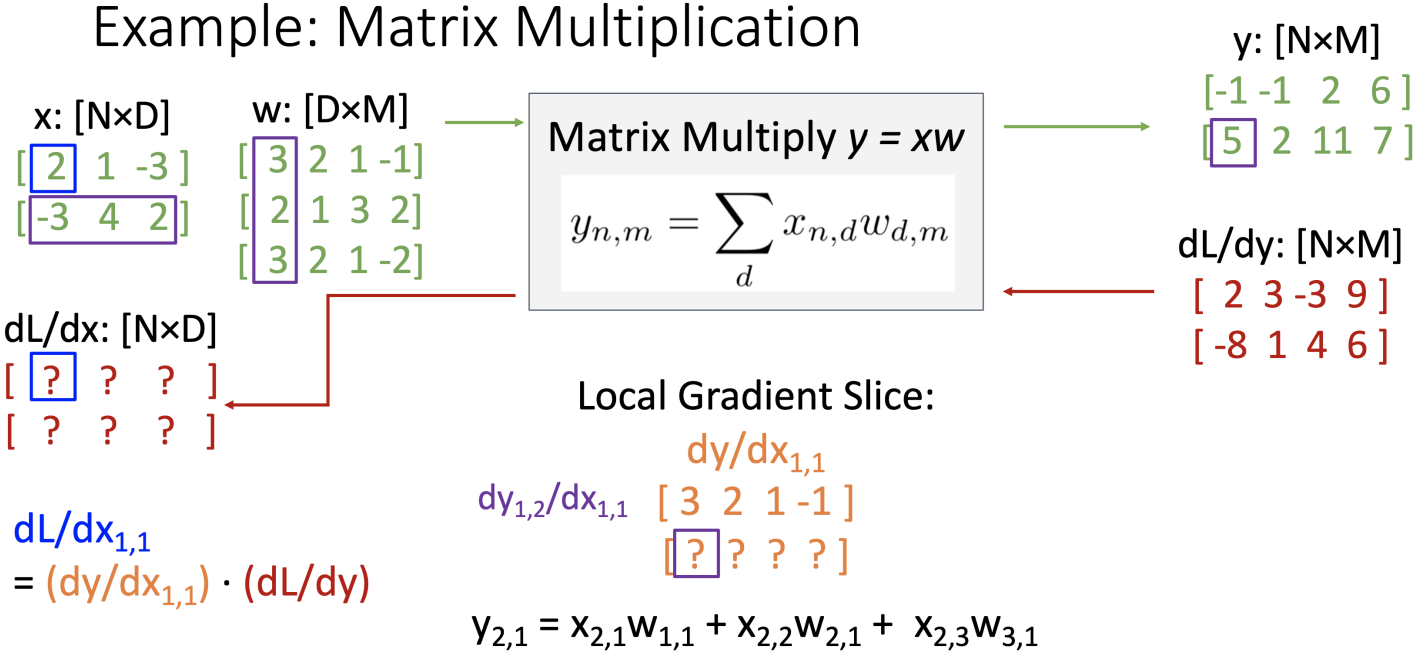
Example: Matrix Multiplication



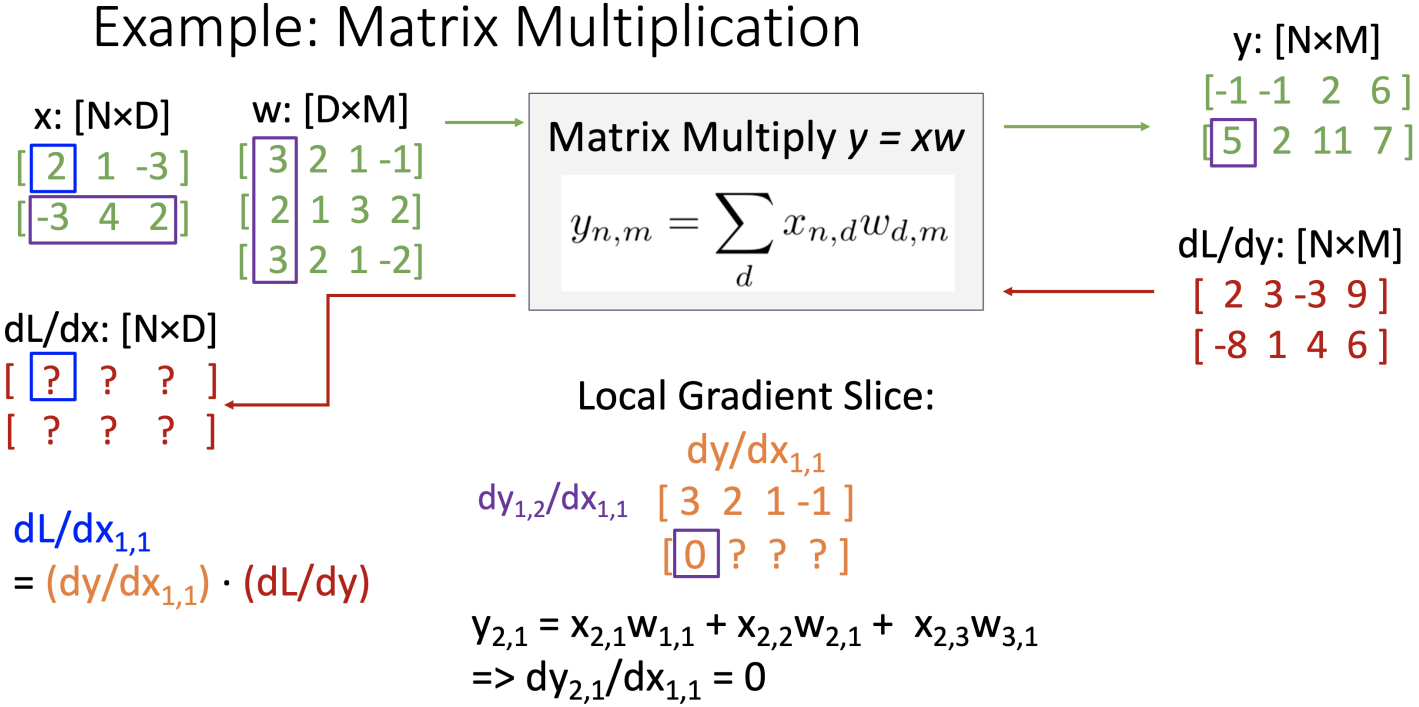
Example: Matrix Multiplication



Example: Matrix Multiplication



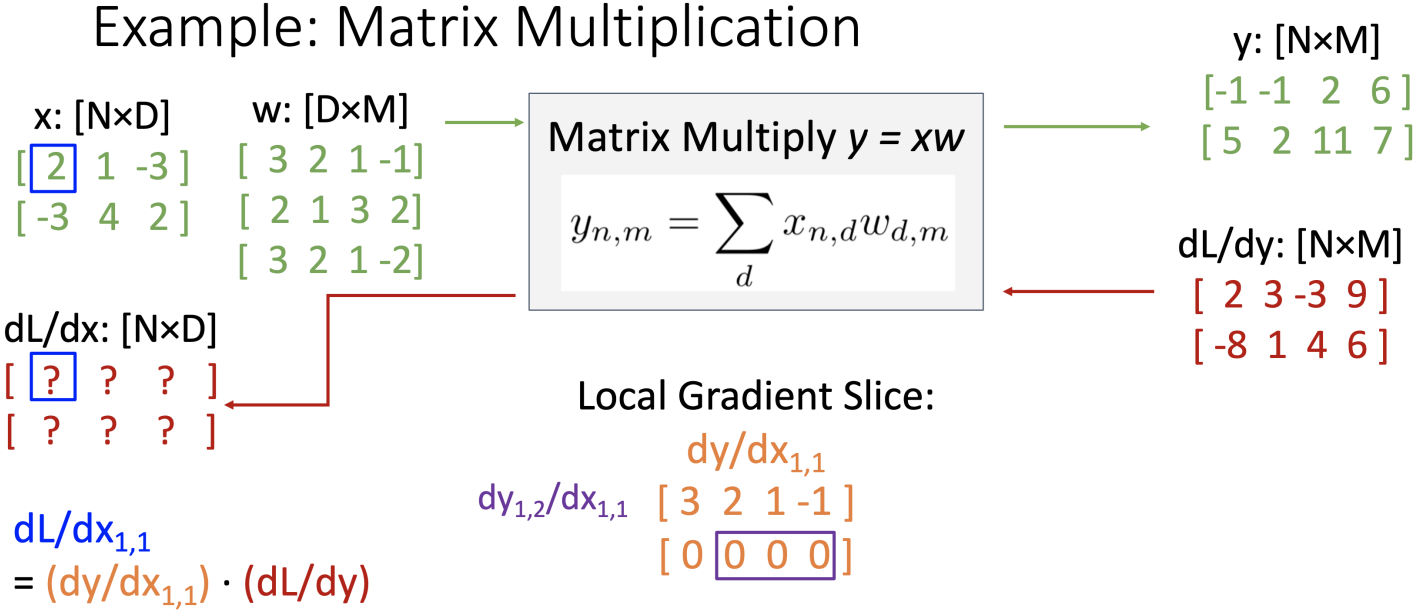
Example: Matrix Multiplication



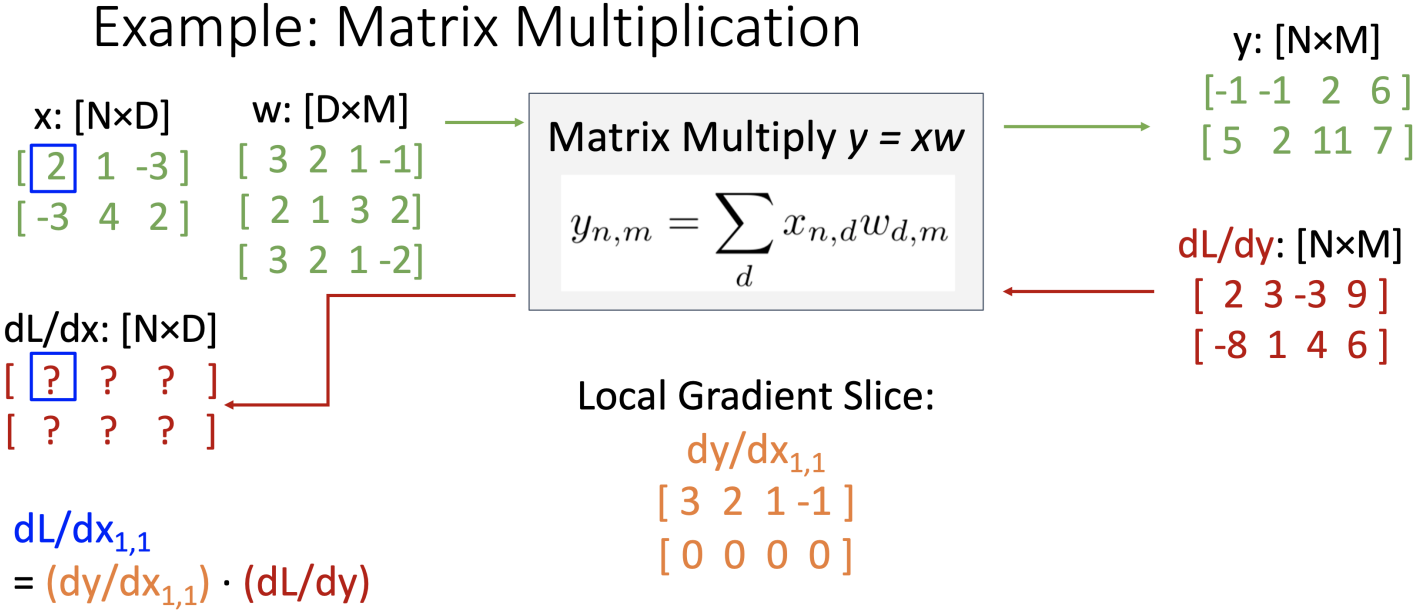
$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$

$y_{2,1} = x_{2,1}w_{1,1} + x_{2,2}w_{2,1} + x_{2,3}w_{3,1}$
$\Rightarrow dy_{2,1}/dx_{1,1} = 0$

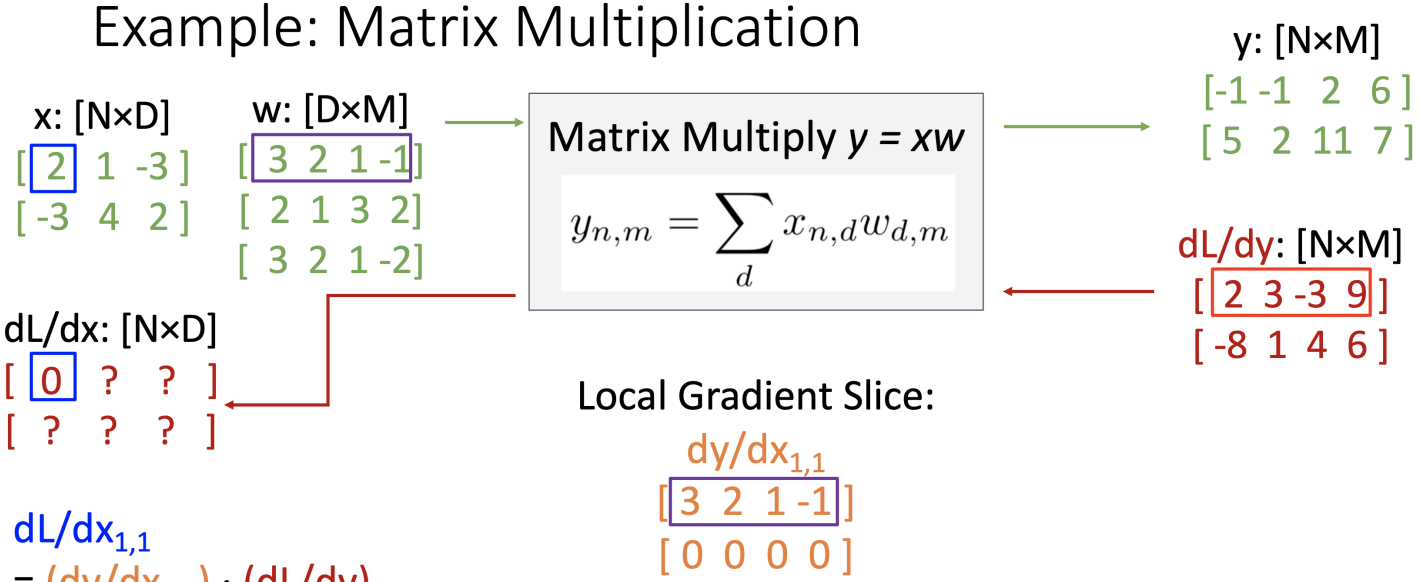
Example: Matrix Multiplication



Example: Matrix Multiplication

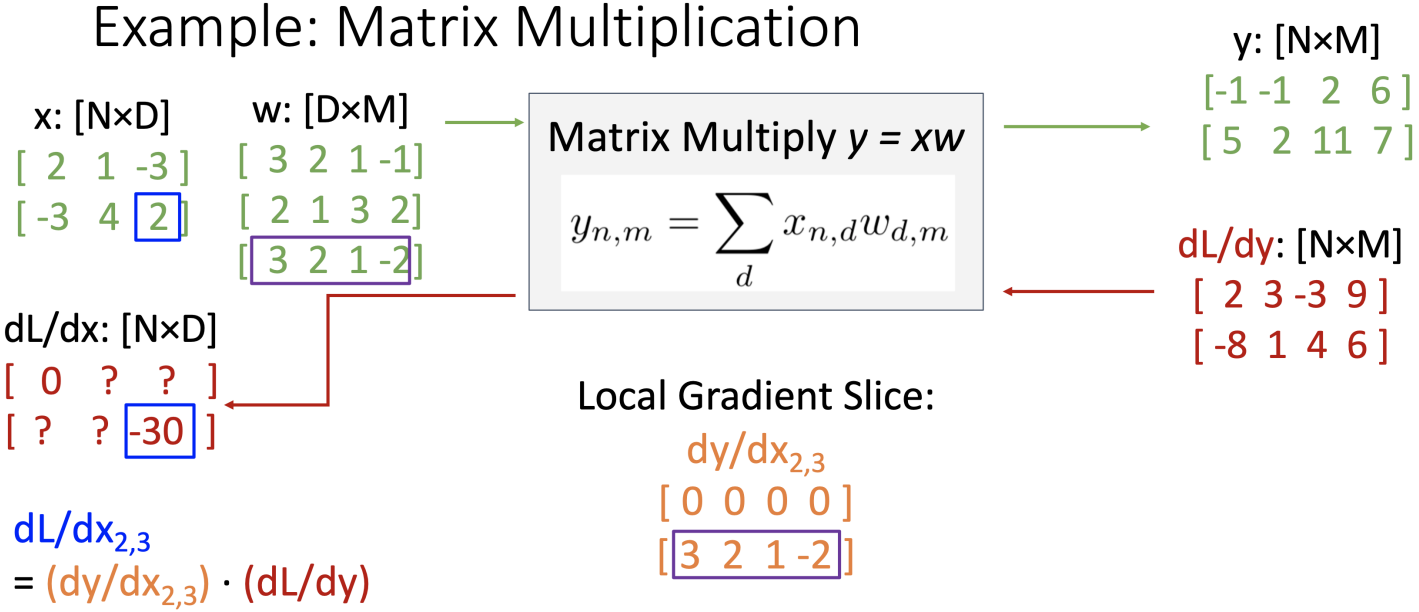


Example: Matrix Multiplication

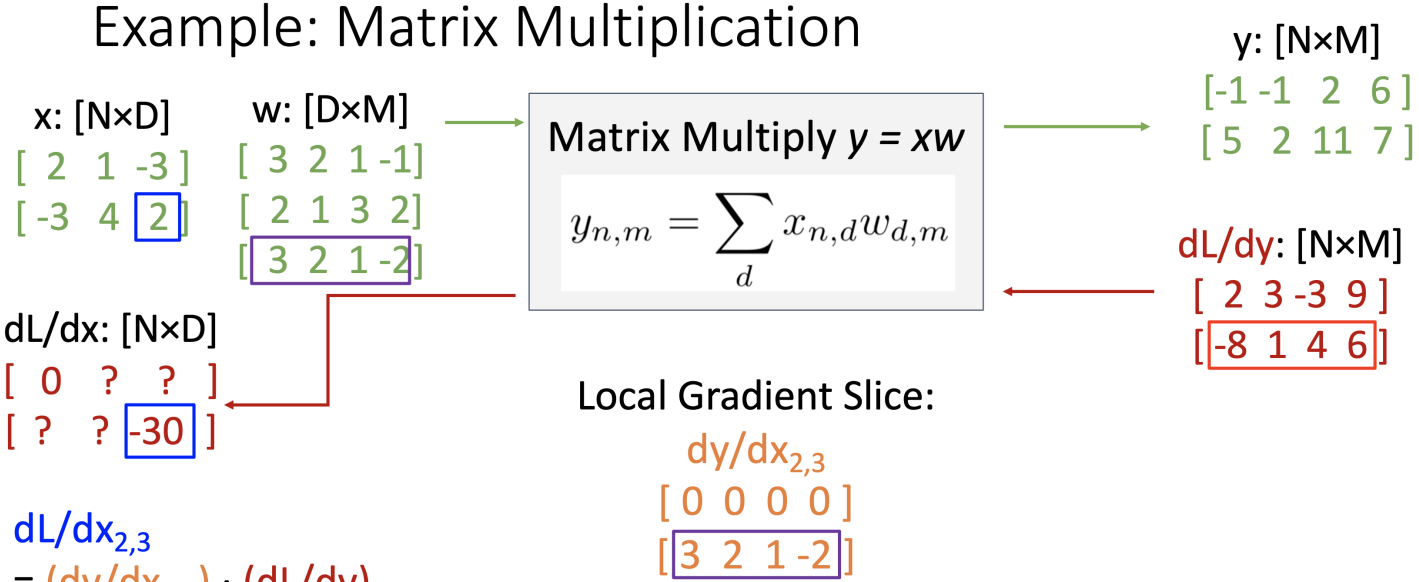


$dL/dx_{1,1}$
$= (dy/dx_{1,1}) \cdot (dL/dy)$
$= (w_{1,:}) \cdot (dL/dy_{1,:})$
$= 3*2 + 2*3 + 1*(-3) + (-1)*9 = 0$

Example: Matrix Multiplication

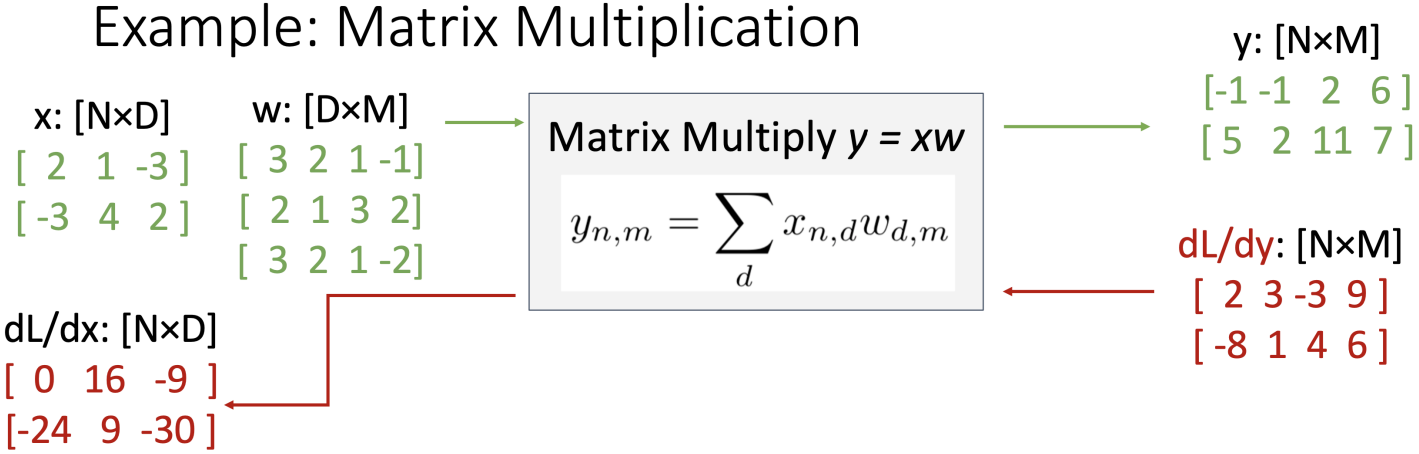


Example: Matrix Multiplication



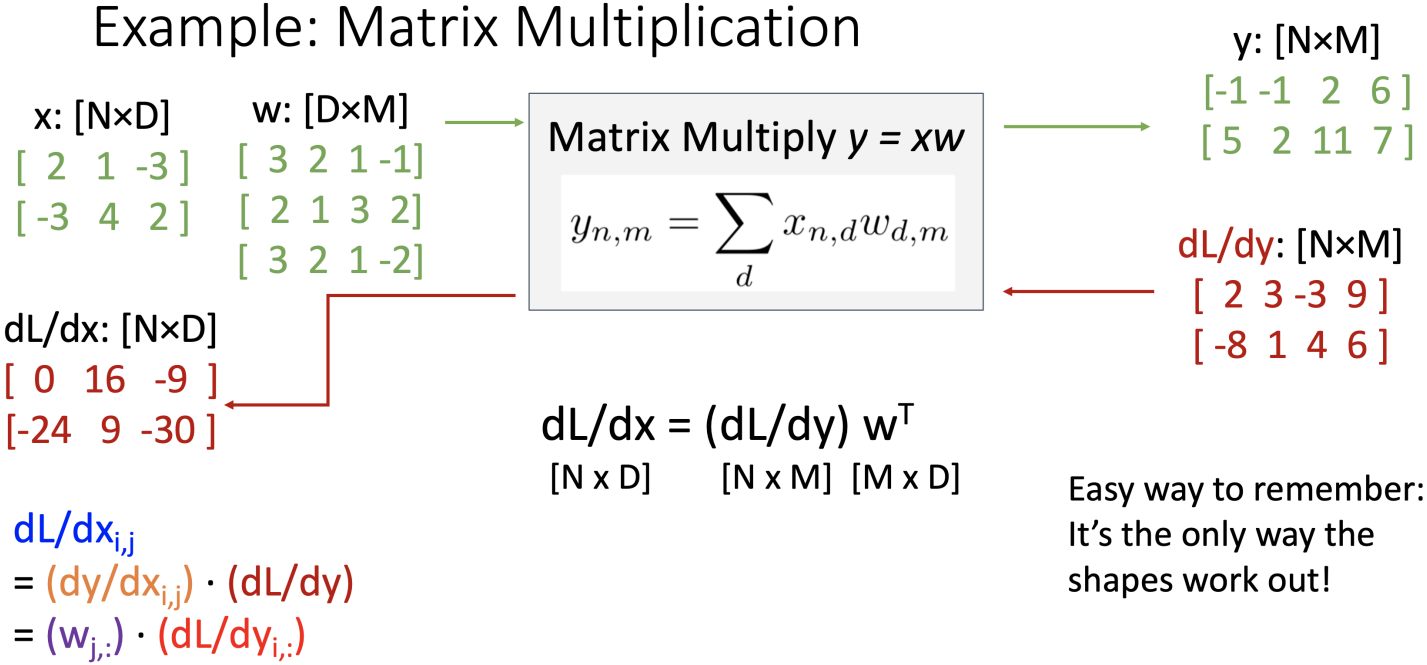
$dL/dx_{2,3}$
$= (dy/dx_{2,3}) \cdot (dL/dy)$
$= (w_{3,:}) \cdot (dL/dy_{2,:})$
$= 3*(-8) + 2*1 + 1*4 + (-2)*6 = -30$

Example: Matrix Multiplication

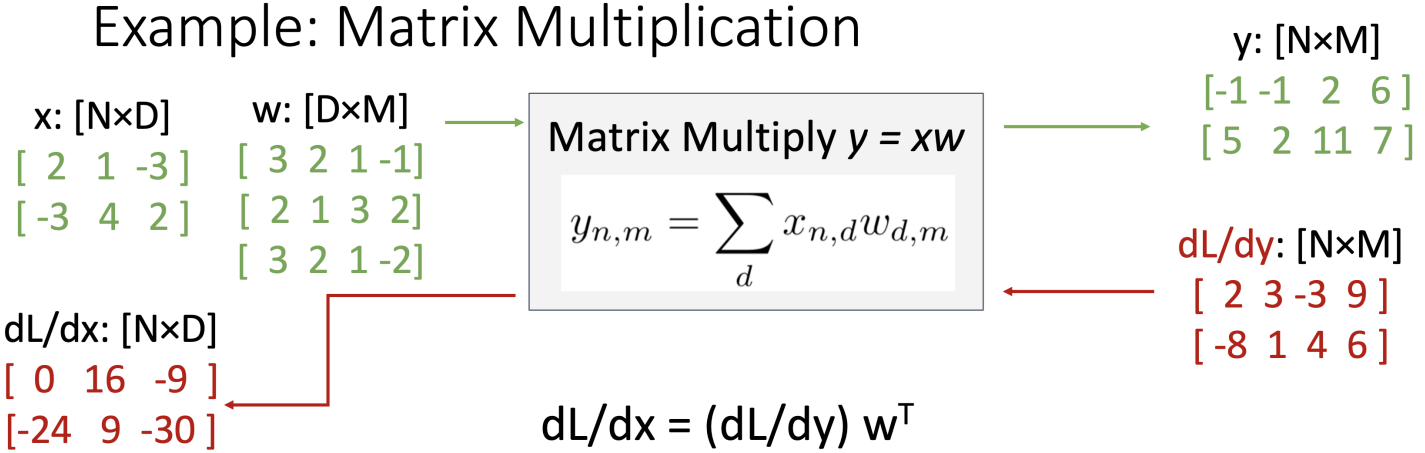


$dL/dx_{i,j}$
$= (dy/dx_{i,j}) \cdot (dL/dy)$
$= (w_{j,:}) \cdot (dL/dy_{i,:})$

Example: Matrix Multiplication



Example: Matrix Multiplication



$dL/dx = (dL/dy)\, w^T$

[N x D] [N x M] [M x D]

$dL/dw = x^T\, (dL/dy)$

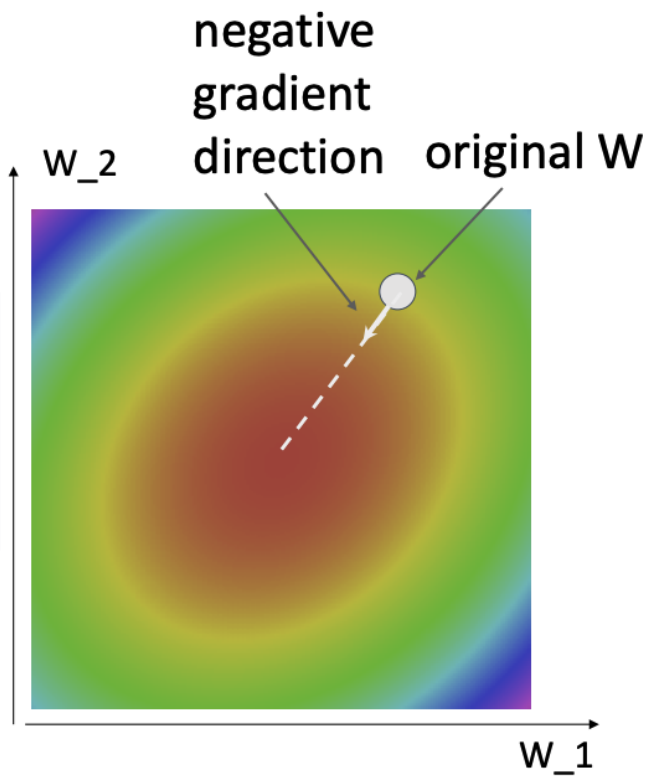
[D x M] [D x N] [N x M]

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

Recall: Optimization $W^* = \arg\min_W L(W)$

- Gradient decent: Iteratively step in the direction of the negative gradient

$$W \leftarrow W - \alpha \nabla_W L(W)$$



Recall: Optimization $L = \frac{1}{N}\sum_{i=1}^{N} \ell_i(\mathbf{x}^{(i)}, , y^{(i)}; W) + \lambda R(W)$

- All the training data is used to perform a single parameter update

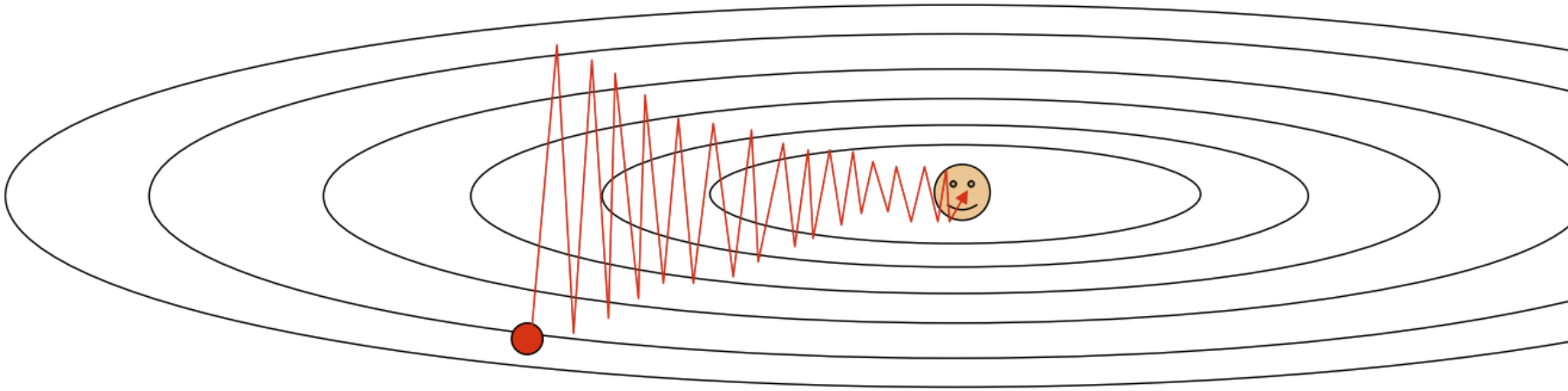
$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W \ell_i(\mathbf{x}^{(i)}, y^{(i)}; W) + \lambda \nabla_W R(W)$$

Full sum is expensive when N is large!

- Mini-batch Gradient Descent: approximate sum using a minibatch of examples; 32/64/128 common
- Stochastic Gradient Descent (SGD): when mini-batch contains only a single example; usually use SGD when referring to mini-batch gradient descent

Problems with SGD

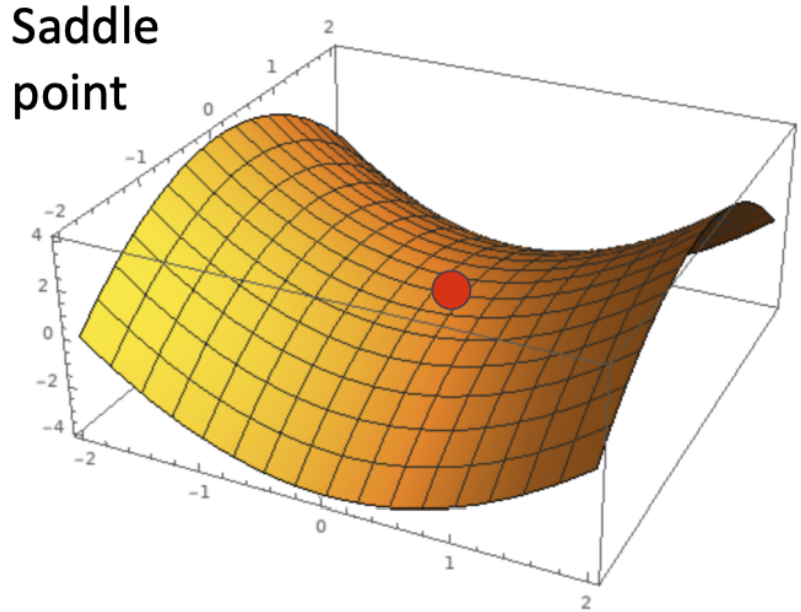
- What if loss changes quickly in one direction and slowly in another? What does gradient decent do?



Very slow progress along shallow direction, jitter along steep direction.

Problems with SGD

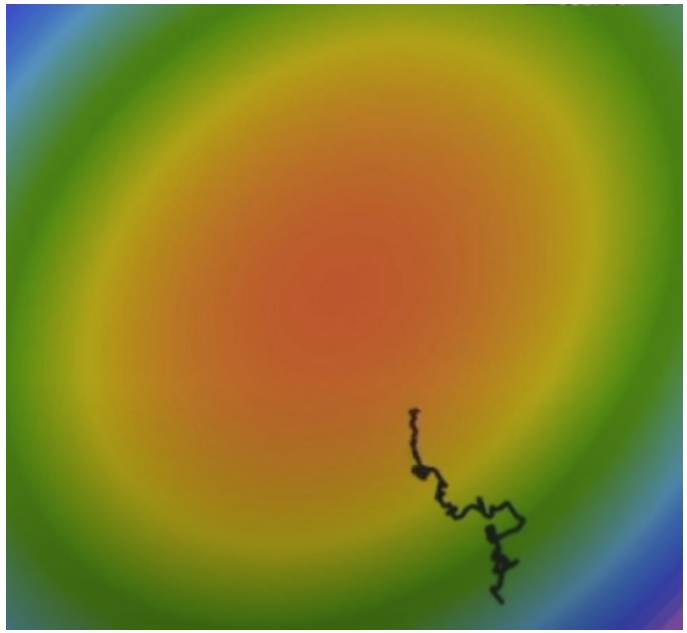
- What if the loss function has a local minimum or saddle point?



Zero gradient, gradient descent gets stuck

Problems with SGD

- Our gradients come from minibatches so they can be noisy!



Use SGD + Momentum instead!
Sutskever et al, “On the importance of initialization and momentum in deep learning", ICML 2013.

Why do you think deep learning became so successful?

Important topics of this lecture

- Deep nets
- Backpropagation

Up next:

- PyTorch Tutorial
- CNN & RNN