

Natural Language Processing with Deep Learning

CS224N/Ling284



Lecture 5:
Backpropagation

Announcements

- Assignment 1 due Thursday, 11:59
 - You can use up to 3 late days (making it due Sunday at midnight)
- Default final project will be released February 1st
- Final project proposal due February 8th
 - See website for details and inspiration

Overview Today:

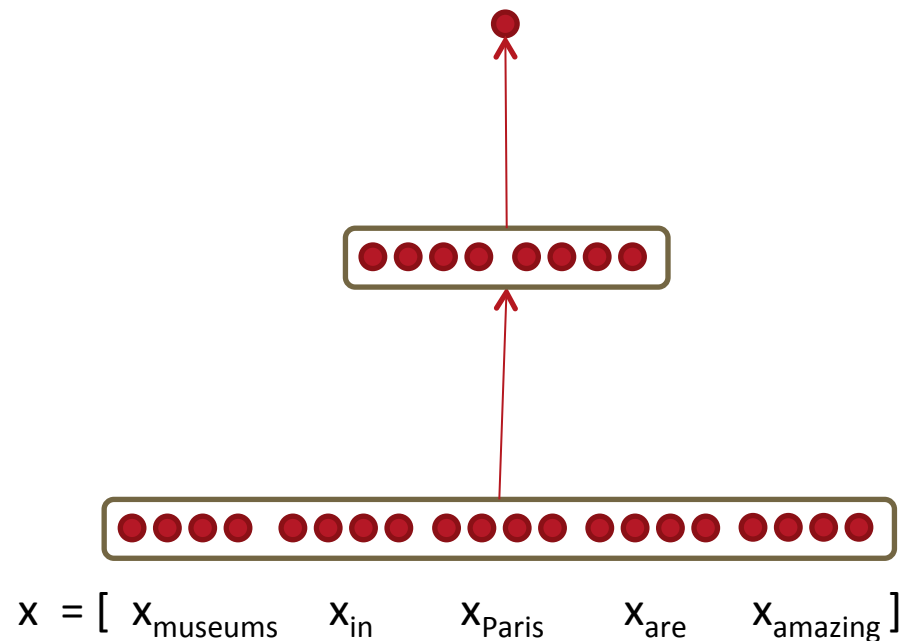
- From one-layer to multi layer neural networks!
- Fully vectorized gradient computation
- The backpropagation algorithm
- (Time permitting) Class project tips

Remember: One-layer Neural Net

$$s = u^T h$$

$$h = f(Wx + b)$$

x (input)



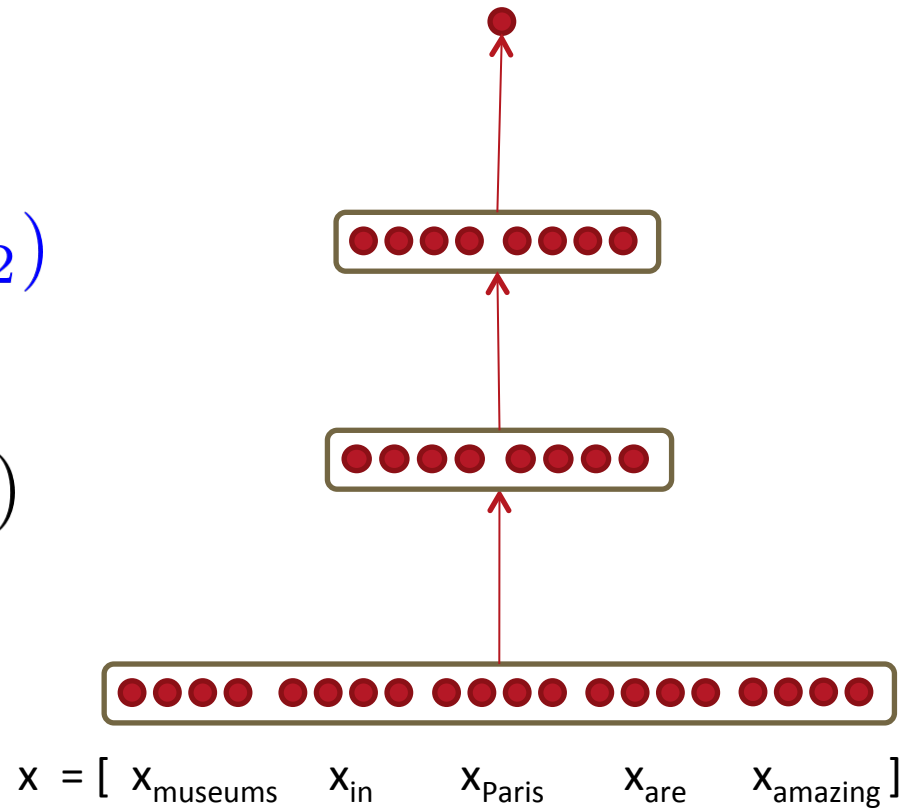
Two-layer Neural Net

$$s = \mathbf{u}^T \mathbf{h}_2$$

$$\mathbf{h}_2 = f(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2)$$

$$\mathbf{h}_1 = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

\mathbf{x} (input)



Repeat as Needed!

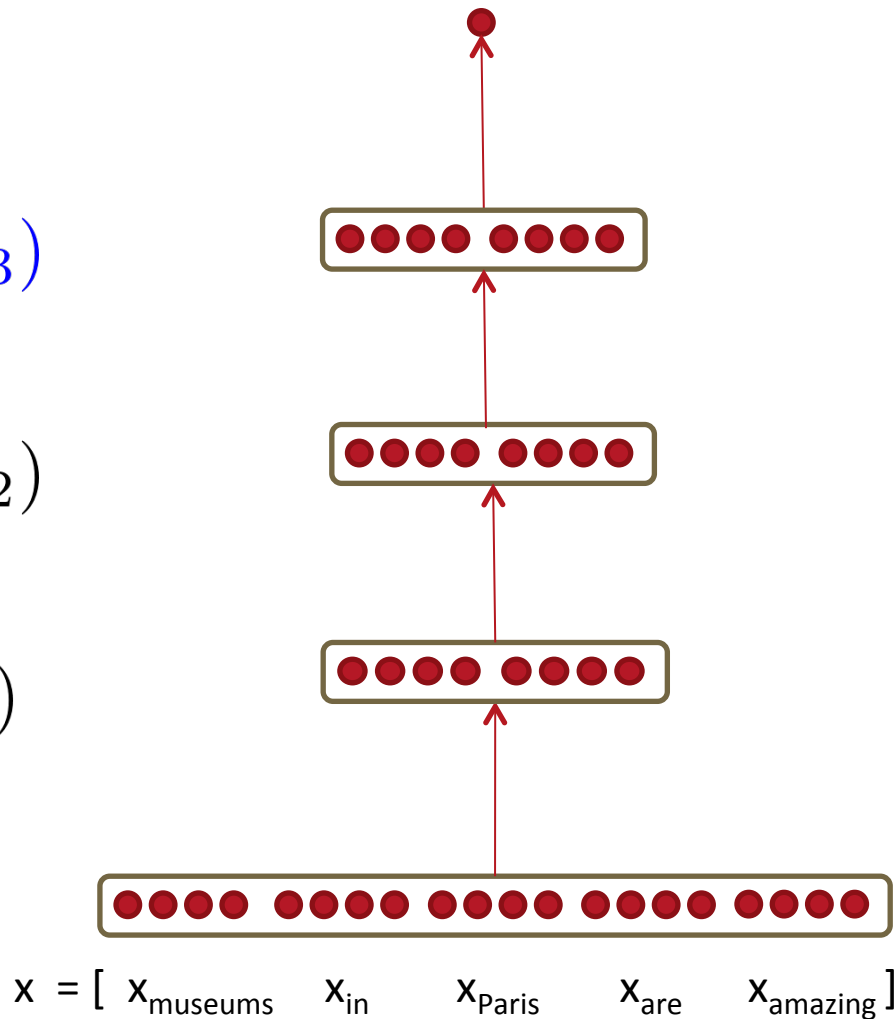
$$s = u^T h_3$$

$$h_3 = f(W_3 h_2 + b_3)$$

$$h_2 = f(W_2 h_1 + b_2)$$

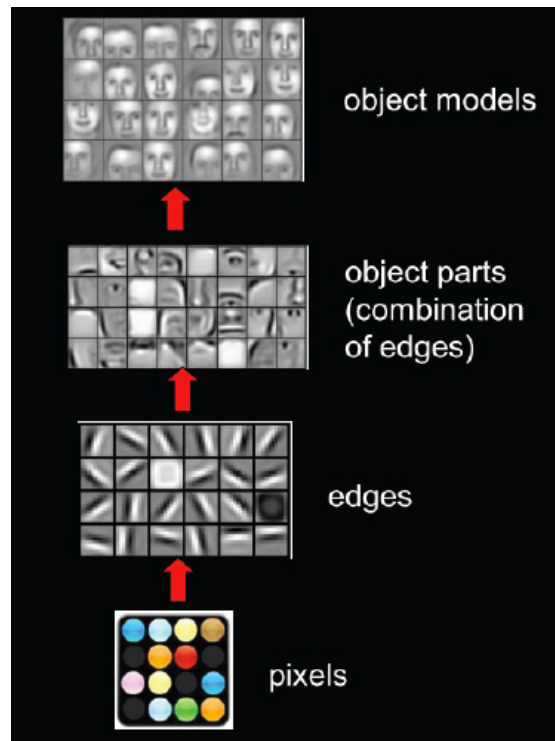
$$h_1 = f(W_1 x + b_1)$$

x (input)



Why Have Multiple Layers?

- Hierarchical representations -> neural net can represent complicated features
- Better results!



# Layers	Machine Translation Score
2	23.7
4	25.3
8	25.5

From Transformer Network (will cover in a later lecture)

Remember: Stochastic Gradient Descent

- Update equation:

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

$\alpha =$ *step size* or *learning rate*

Remember: Stochastic Gradient Descent

- Update equation:

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

$\alpha =$ *step size* or *learning rate*

- This Lecture: How do we compute $\nabla_{\theta} J(\theta)$?
 - By hand
 - Algorithmically (the backpropagation algorithm)

Why learn all these details about gradients?

- Modern deep learning frameworks compute gradients for you
- But why take a class on compilers or systems when they are implemented for you?
 - Understanding what is going on under the hood is useful!
- Backpropagation doesn't always work perfectly.
 - Understanding why is crucial for debugging and improving models
 - Example in future lecture: exploding and vanishing gradients

Quickly Computing Gradients by Hand

- Review of multivariable derivatives
- Fully vectorized gradients
 - Much faster and more useful than non-vectorized gradients
 - But doing a non-vectorized gradient can be good practice, see slides in last week's lecture for an example
 - Lecture notes cover this material in more detail

Gradients

- Given a function with 1 output and n inputs

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$$

- It's gradient is a vector of partial derivatives

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

Jacobian Matrix: Generalization of the Gradient

- Given a function with **m outputs** and n inputs

$$\mathbf{f}(\mathbf{x}) = [f_1(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n)]$$

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

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

Chain Rule For Jacobians

- For one-variable functions: multiply derivatives

$$z = 3y$$

$$y = x^2$$

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} = (3)(2x) = 6x$$

- For multiple variables: multiply Jacobians

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \dots$$

Example Jacobian: Activation Function

$$\mathbf{h} = f(\mathbf{z}), \text{ what is } \frac{\partial \mathbf{h}}{\partial \mathbf{z}}?$$

$$\mathbf{h}, \mathbf{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

Example Jacobian: Activation Function

$$\mathbf{h} = f(\mathbf{z}), \text{ what is } \frac{\partial \mathbf{h}}{\partial \mathbf{z}}? \quad \mathbf{h}, \mathbf{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

Function has n outputs and n inputs $\rightarrow n$ by n Jacobian

Example Jacobian: Activation Function

$$\mathbf{h} = f(\mathbf{z}), \text{ what is } \frac{\partial \mathbf{h}}{\partial \mathbf{z}}?$$

$$\mathbf{h}, \mathbf{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}} \right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i)$$

definition of Jacobian

Example Jacobian: Activation Function

$$\mathbf{h} = f(\mathbf{z}), \text{ what is } \frac{\partial \mathbf{h}}{\partial \mathbf{z}}?$$

$$\mathbf{h}, \mathbf{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\begin{aligned} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}} \right)_{ij} &= \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \\ &= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \end{aligned}$$

definition of Jacobian

regular 1-variable derivative

Example Jacobian: Activation Function

$$\mathbf{h} = f(\mathbf{z}), \text{ what is } \frac{\partial \mathbf{h}}{\partial \mathbf{z}}?$$

$$\mathbf{h}, \mathbf{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}} \right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i)$$

definition of Jacobian

$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}$$

regular 1-variable derivative

$$\frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \begin{pmatrix} f'(z_1) & & 0 \\ & \ddots & \\ 0 & & f'(z_n) \end{pmatrix} = \text{diag}(\mathbf{f}'(\mathbf{z}))$$

Other Jacobians

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{W}$$

Other Jacobians

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{W}$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I} \text{ (Identity matrix)}$$

Other Jacobians

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{W}$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I} \text{ (Identity matrix)}$$

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

Other Jacobians

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{W}$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I} \text{ (Identity matrix)}$$

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

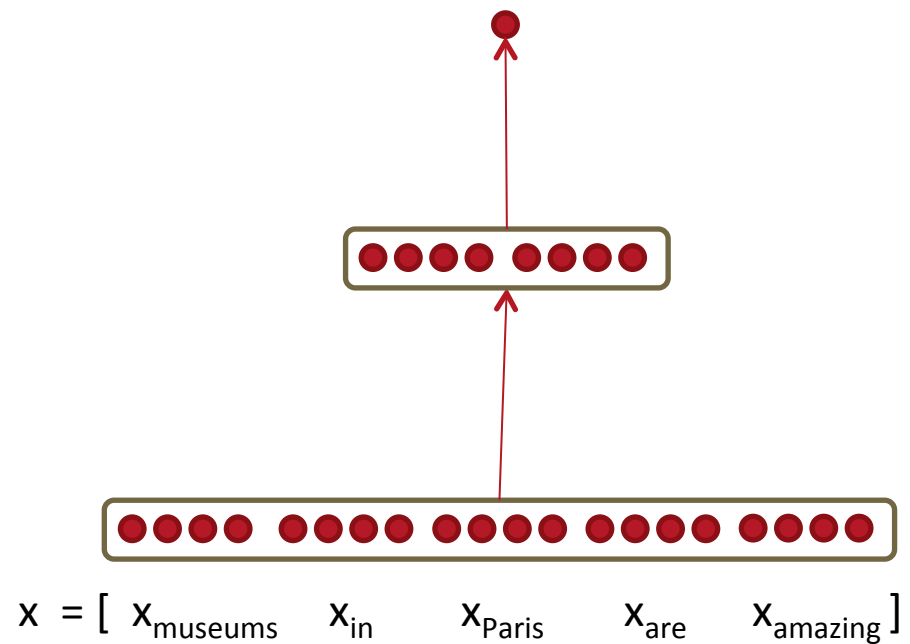
- Compute these at home for practice!
 - Check your answers with the lecture notes

Back to Neural Nets!

$$s = u^T h$$

$$h = f(Wx + b)$$

x (input)



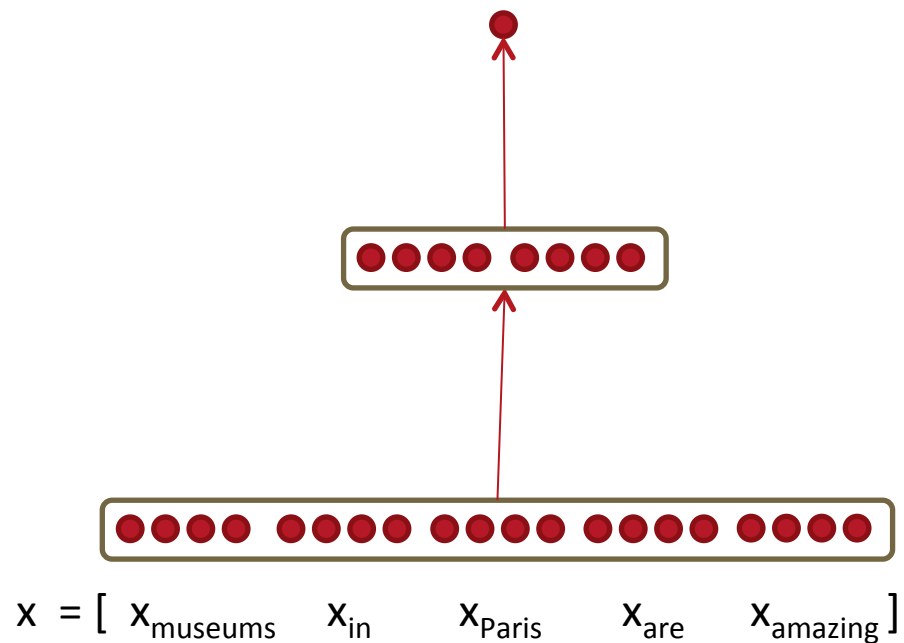
Back to Neural Nets!

- Let's find $\frac{\partial s}{\partial b}$
 - In practice we care about the gradient of the loss, but we will compute the gradient of the score for simplicity

$$s = u^T h$$

$$h = f(Wx + b)$$

x (input)



1. Break up equations into simple pieces

$$s = u^T h$$

$$s = u^T h$$

$$h = f(Wx + b)$$



$$h = f(\mathbf{z})$$

$$\mathbf{z} = Wx + b$$

$$x \quad (\text{input})$$

$$x \quad (\text{input})$$

2. Apply the chain rule

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} \quad (\text{input})$$

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

2. Apply the chain rule

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} \quad (\text{input})$$

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

2. Apply the chain rule

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

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

2. Apply the chain rule

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

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

3. Write out the Jacobians

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} \quad (\text{input})$$

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

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

$$\frac{\partial}{\partial \mathbf{z}} (f(\mathbf{z})) = \text{diag}(f'(\mathbf{z}))$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I}$$

3. Write out the Jacobians

$$s = u^T h$$

$$h = f(z)$$

$$z = \mathbf{W}x + b$$

$$x \text{ (input)}$$

$$\begin{aligned} \frac{\partial s}{\partial b} &= \frac{\partial s}{\partial h} \frac{\partial h}{\partial z} \frac{\partial z}{\partial b} \\ &\quad \downarrow \\ &= h^T \end{aligned}$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial u}(u^T h) = h^T$$

$$\frac{\partial}{\partial z}(f(z)) = \text{diag}(f'(z))$$

$$\frac{\partial}{\partial b}(\mathbf{W}x + b) = \mathbf{I}$$

3. Write out the Jacobians

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

$$\begin{aligned} \frac{\partial s}{\partial \mathbf{b}} &= \frac{\partial s}{\partial \mathbf{h}} \quad \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \quad \frac{\partial \mathbf{z}}{\partial \mathbf{b}} \\ &\quad \downarrow \quad \downarrow \\ &= \mathbf{h}^T \text{diag}(f'(\mathbf{z})) \end{aligned}$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

$$\frac{\partial}{\partial \mathbf{z}} (f(\mathbf{z})) = \text{diag}(f'(\mathbf{z}))$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I}$$

3. Write out the Jacobians

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

$$\begin{aligned} \frac{\partial s}{\partial \mathbf{b}} &= \frac{\partial s}{\partial \mathbf{h}} \quad \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \quad \frac{\partial \mathbf{z}}{\partial \mathbf{b}} \\ &\quad \downarrow \quad \downarrow \quad \downarrow \\ &= \mathbf{h}^T \text{diag}(f'(\mathbf{z})) \mathbf{I} \end{aligned}$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

$$\frac{\partial}{\partial \mathbf{z}} (f(\mathbf{z})) = \text{diag}(f'(\mathbf{z}))$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I}$$

3. Write out the Jacobians

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} \quad (\text{input})$$

$$\begin{aligned} \frac{\partial s}{\partial \mathbf{b}} &= \frac{\partial s}{\partial \mathbf{h}} \quad \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \quad \frac{\partial \mathbf{z}}{\partial \mathbf{b}} \\ &\quad \downarrow \quad \downarrow \quad \downarrow \\ &= \mathbf{h}^T \text{diag}(f'(\mathbf{z})) \mathbf{I} \\ &= \mathbf{h}^T \circ f'(\mathbf{z}) \end{aligned}$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \mathbf{u}} (\mathbf{u}^T \mathbf{h}) = \mathbf{h}^T$$

$$\frac{\partial}{\partial \mathbf{z}} (f(\mathbf{z})) = \text{diag}(f'(\mathbf{z}))$$

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I}$$

Re-using Computation

- Suppose we now want to compute $\frac{\partial s}{\partial \mathbf{W}}$
 - Using the chain rule again:

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

Re-using Computation

- Suppose we now want to compute $\frac{\partial s}{\partial \mathbf{W}}$
 - Using the chain rule again:

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

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

The same! Let's avoid duplicated computation...

Re-using Computation

- Suppose we now want to compute $\frac{\partial s}{\partial \mathbf{W}}$
 - Using the chain rule again:

$$\frac{\partial s}{\partial \mathbf{W}} = \delta \frac{\partial z}{\partial \mathbf{W}}$$

$$\frac{\partial s}{\partial \mathbf{b}} = \delta \frac{\partial z}{\partial \mathbf{b}} = \delta$$

$$\delta = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \mathbf{h}^T \circ f'(\mathbf{z})$$

Derivative with respect to Matrix

- What does $\frac{\partial s}{\partial \mathbf{W}}$ look like? $\mathbf{W} \in \mathbb{R}^{n \times m}$
- 1 output, nm inputs: 1 by nm Jacobian?
 - Inconvenient to do $\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$

Derivative with respect to Matrix

- What does $\frac{\partial s}{\partial \mathbf{W}}$ look like? $\mathbf{W} \in \mathbb{R}^{n \times m}$
- 1 output, nm inputs: 1 by nm Jacobian?
 - Inconvenient to do $\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$
- Instead follow convention: shape of the gradient is shape of parameters

- So $\frac{\partial s}{\partial \mathbf{W}}$ is n by m :
$$\begin{bmatrix} \frac{\partial s}{\partial W_{11}} & \cdots & \frac{\partial s}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial s}{\partial W_{n1}} & \cdots & \frac{\partial s}{\partial W_{nm}} \end{bmatrix}$$

Derivative with respect to Matrix

- Remember $\frac{\partial s}{\partial \mathbf{W}} = \delta \frac{\partial z}{\partial \mathbf{W}}$
 - δ is going to be in our answer
 - The other term should be \mathbf{x} because $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- It turns out $\frac{\partial s}{\partial \mathbf{W}} = \delta^T \mathbf{x}^T$

Why the Transposes?

$$\frac{\partial s}{\partial \mathbf{W}} = \boldsymbol{\delta}^T \mathbf{x}^T$$
$$[n \times m] \quad [n \times 1][1 \times m]$$

- Hacky answer: this makes the dimensions work out
 - Useful trick for checking your work!
- Full explanation in the lecture notes

Why the Transposes?

$$\frac{\partial s}{\partial \mathbf{W}} = \boldsymbol{\delta}^T \mathbf{x}^T = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \end{bmatrix} [x_1, \dots, x_m] = \begin{bmatrix} \delta_1 x_1 & \dots & \delta_1 x_m \\ \vdots & \ddots & \vdots \\ \delta_n x_1 & \dots & \delta_n x_m \end{bmatrix}$$

- Hacky answer: this makes the dimensions work out
 - Useful trick for checking your work!
- Full explanation in the lecture notes

What shape should derivatives be?

- $\frac{\partial s}{\partial \mathbf{b}} = \mathbf{h}^T \circ f'(z)$ is a row vector
 - But convention says our gradient should be a column vector because \mathbf{b} is a column vector...
- Disagreement between Jacobian form (which makes the chain rule easy) and the shape convention (which makes implementing SGD easy)
 - We expect answers to follow the shape convention
 - But Jacobian form is useful for computing the answers

What shape should derivatives be?

- Two options:
- 1. Use Jacobian form as much as possible, reshape to follow the convention at the end:
 - What we just did. But at the end transpose $\frac{\partial s}{\partial \mathbf{b}}$ to make the derivative a column vector, resulting in δ^T
- 2. Always follow the convention
 - Look at dimensions to figure out when to transpose and/or reorder terms.

Notes on PA1

- Don't worry if you used some other method for gradient computation (as long as your answer is right and you are consistent!)
- This lecture we computed the gradient with respect to the score, but in PA1 it's with the loss
- Don't forget to replace f' with the actual derivative
- PA1 uses $x\mathbf{W} + \mathbf{b}$ for the linear transformation: gradients are different!

Backpropagation

- Compute gradients algorithmically
- Converting what we just did by hand into an algorithm
- Used by deep learning frameworks (TensorFlow, PyTorch, etc.)

Computational Graphs

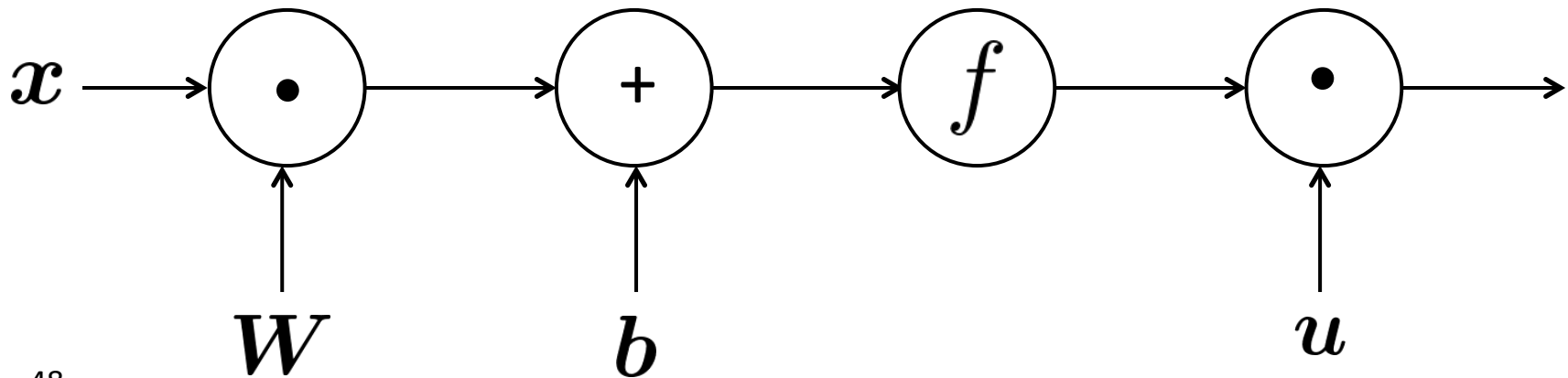
- Representing our neural net equations as a graph
 - Source nodes: inputs
 - Interior nodes: operations

$$s = u^T h$$

$$h = f(z)$$

$$z = \mathbf{W}x + b$$

$$x \quad (\text{input})$$



Computational Graphs

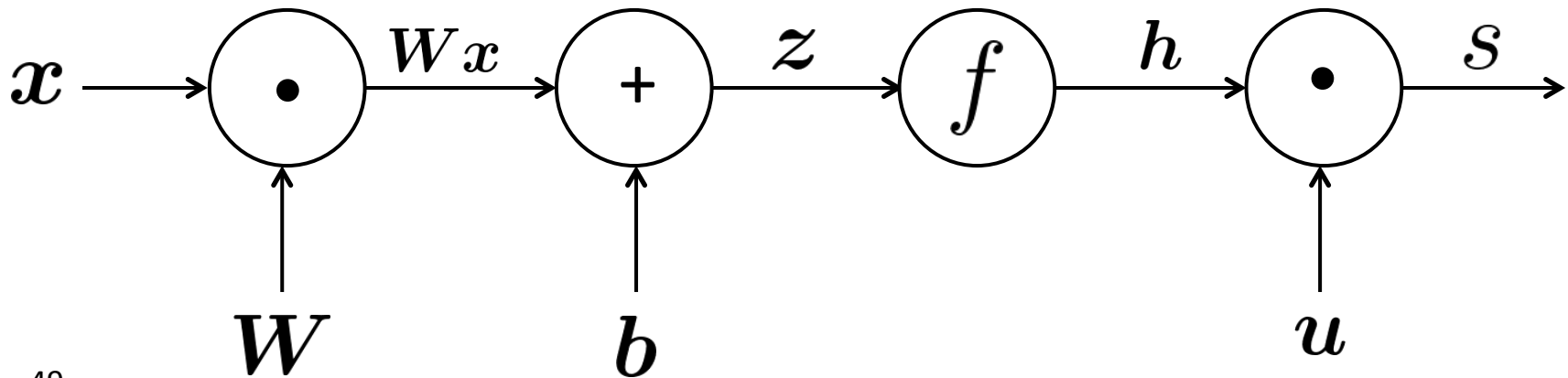
- Representing our neural net equations as a graph
 - Source nodes: inputs
 - Interior nodes: operations
 - Edges pass along result of the operation

$$s = u^T h$$

$$h = f(z)$$

$$z = Wx + b$$

$$x \quad (\text{input})$$



Computational Graphs

- Representing our neural net equations as a graph

$$s = u^T h$$

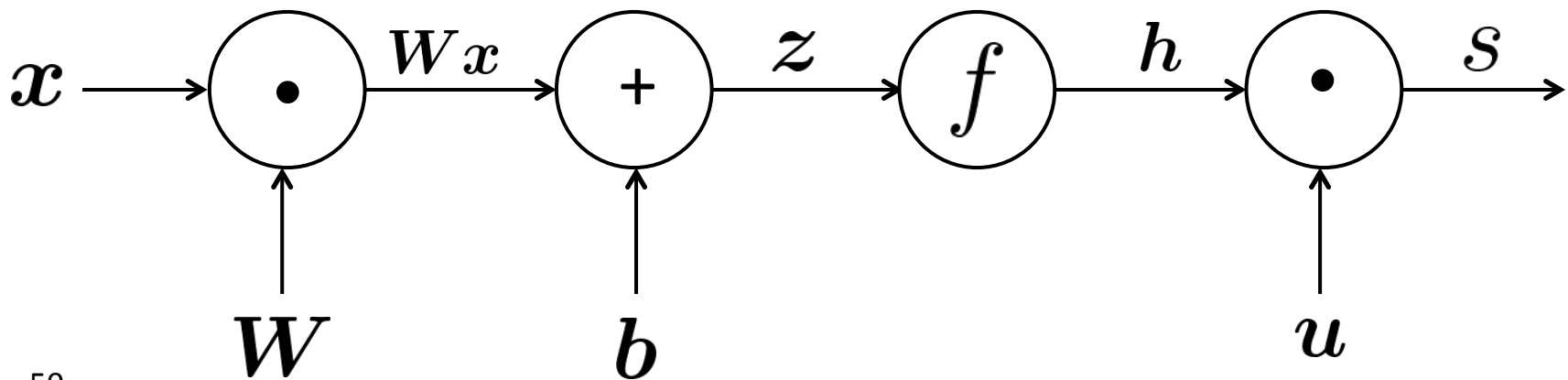
$$h = f(z)$$

$$z = Wx + b$$

(output)

“Forward Propagation”

operation



Backpropagation

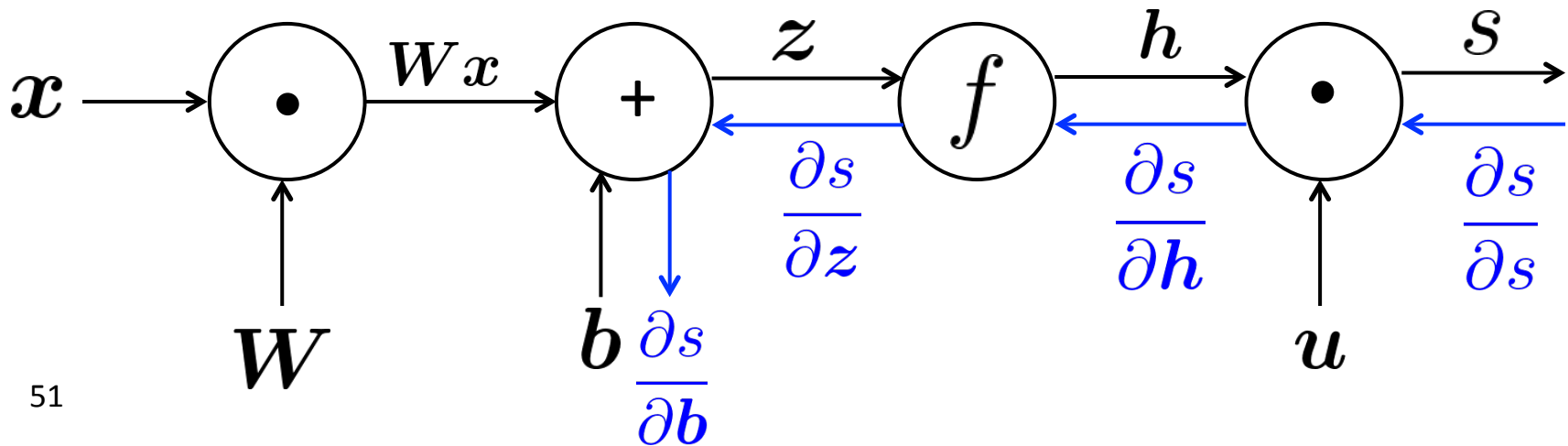
- Go backwards along edges
 - Pass along **gradients**

$$s = u^T h$$

$$h = f(z)$$

$$z = Wx + b$$

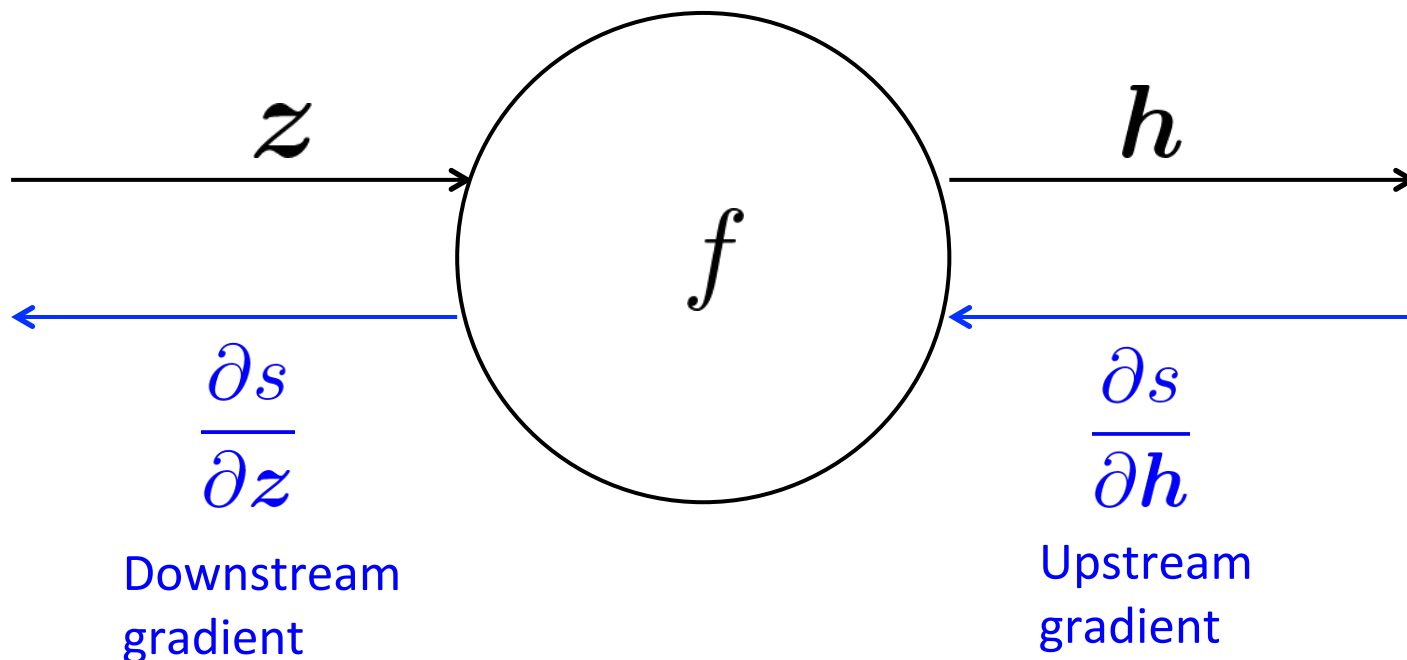
$$x \text{ (input)}$$



Backpropagation: Single Node

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

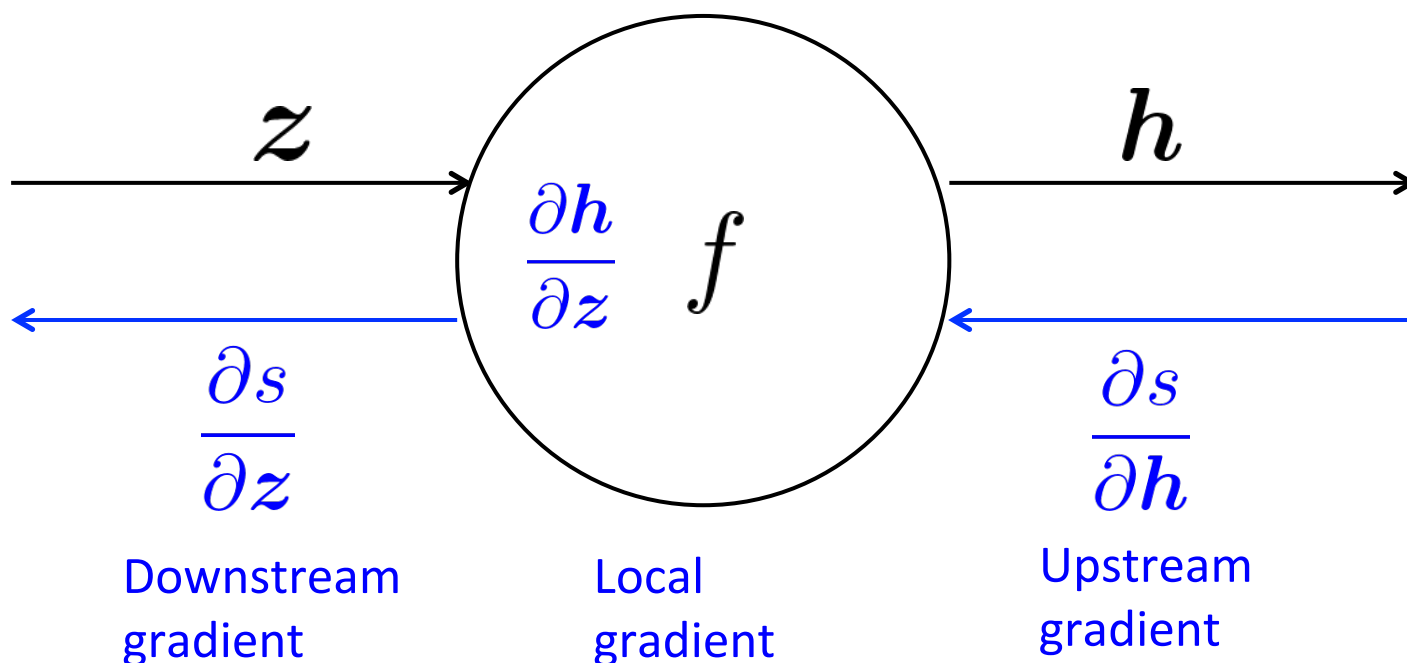
$$h = f(z)$$



Backpropagation: Single Node

- Each node has a **local gradient**
 - The gradient of it's output with respect to it's input

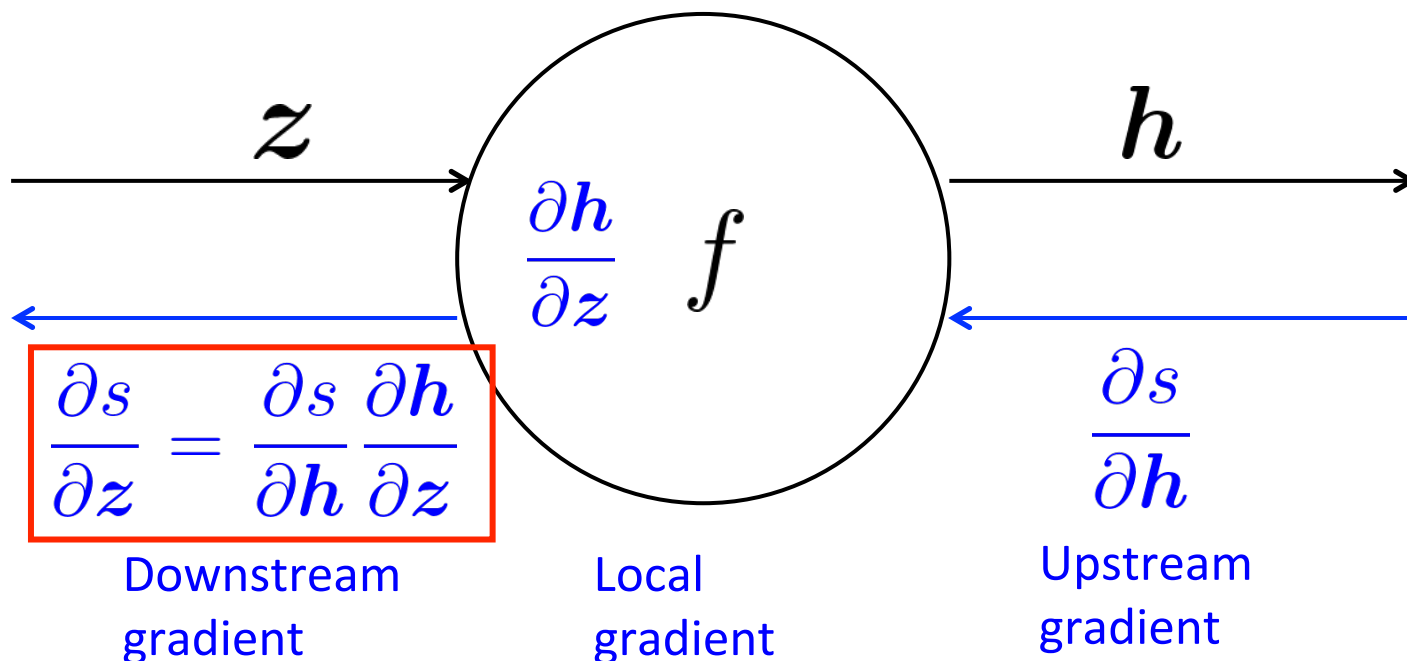
$$h = f(z)$$



Backpropagation: Single Node

- Each node has a **local gradient**
 - The gradient of it's output with respect to it's input

$$h = f(z)$$

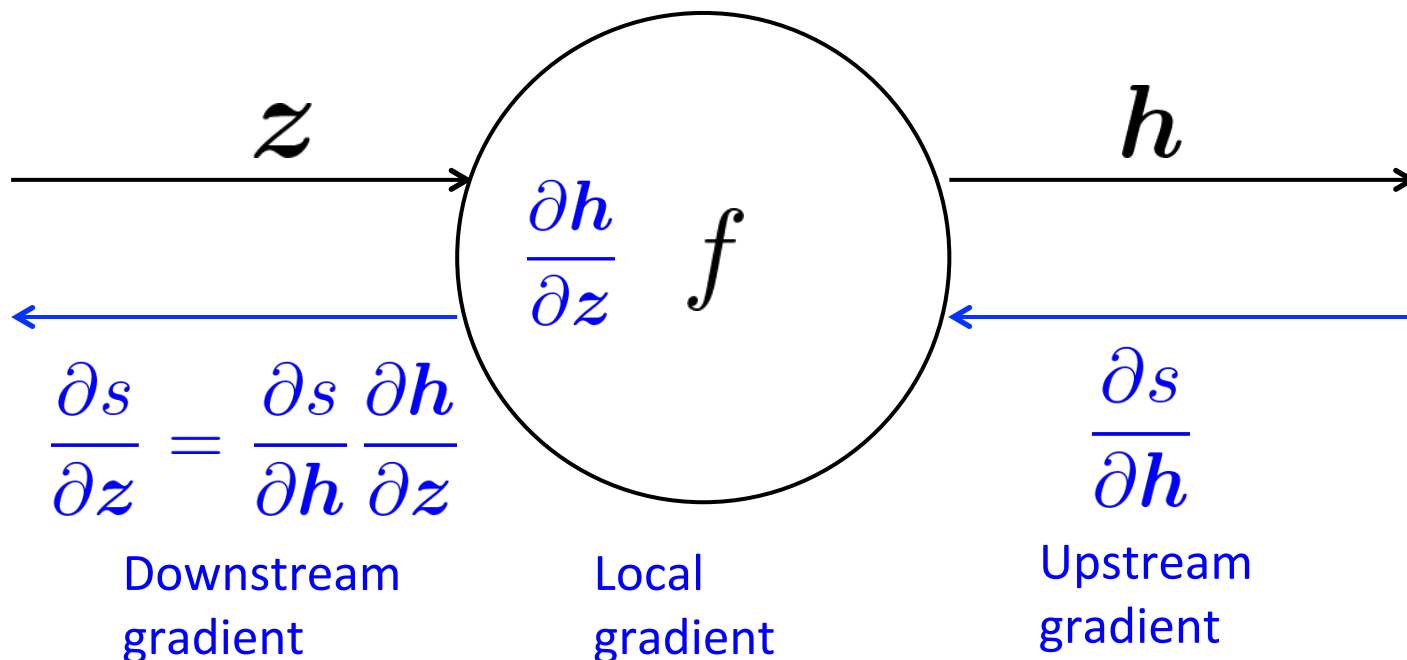


Backpropagation: Single Node

- Each node has a **local gradient**
 - The gradient of it's output with respect to it's input

$$h = f(z)$$

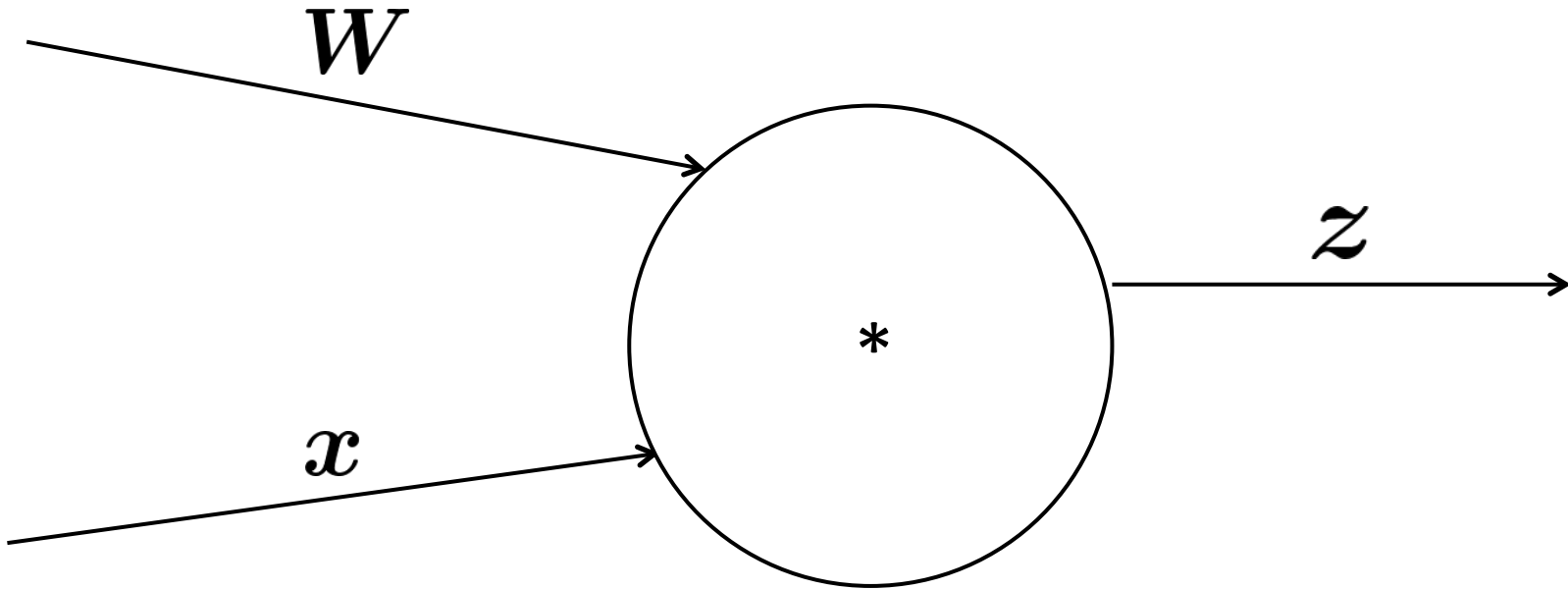
- [downstream gradient] = [upstream gradient] x [local gradient]



Backpropagation: Single Node

- What about nodes with multiple inputs?

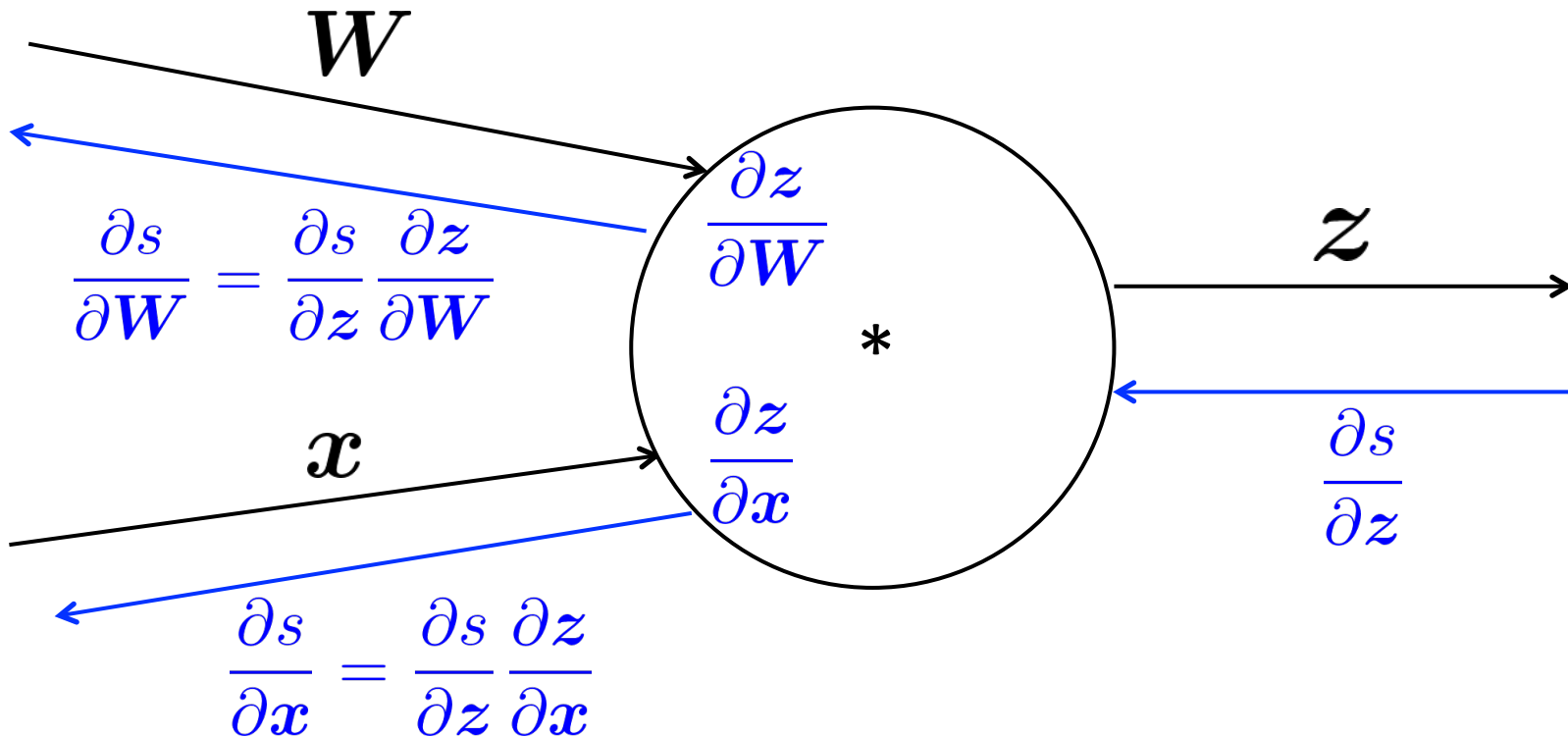
$$z = Wx$$



Backpropagation: Single Node

- Multiple inputs -> multiple local gradients

$$z = Wx$$



Downstream
gradients

Local
gradients

Upstream
gradient

An Example

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

An Example

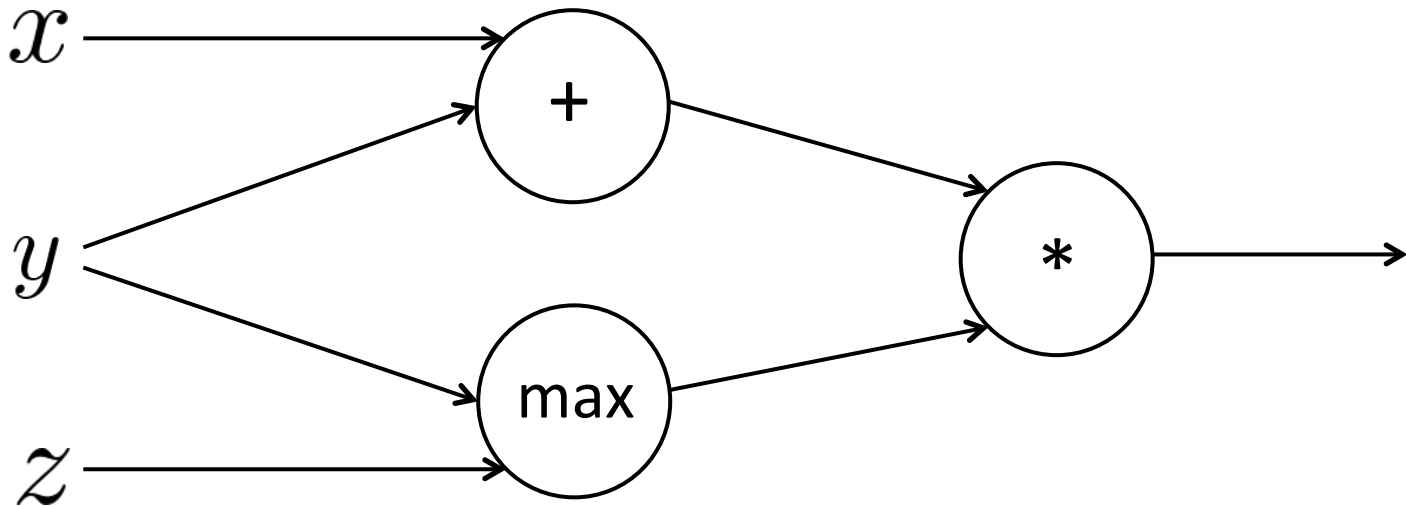
$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



An Example

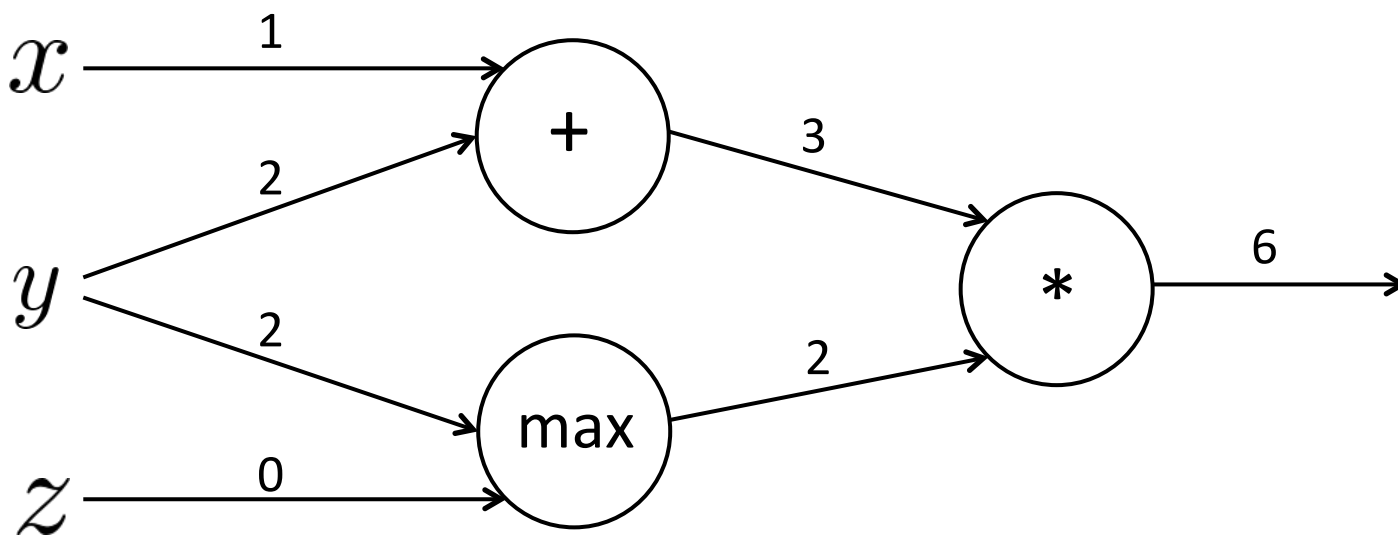
$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



An Example

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

Forward prop steps

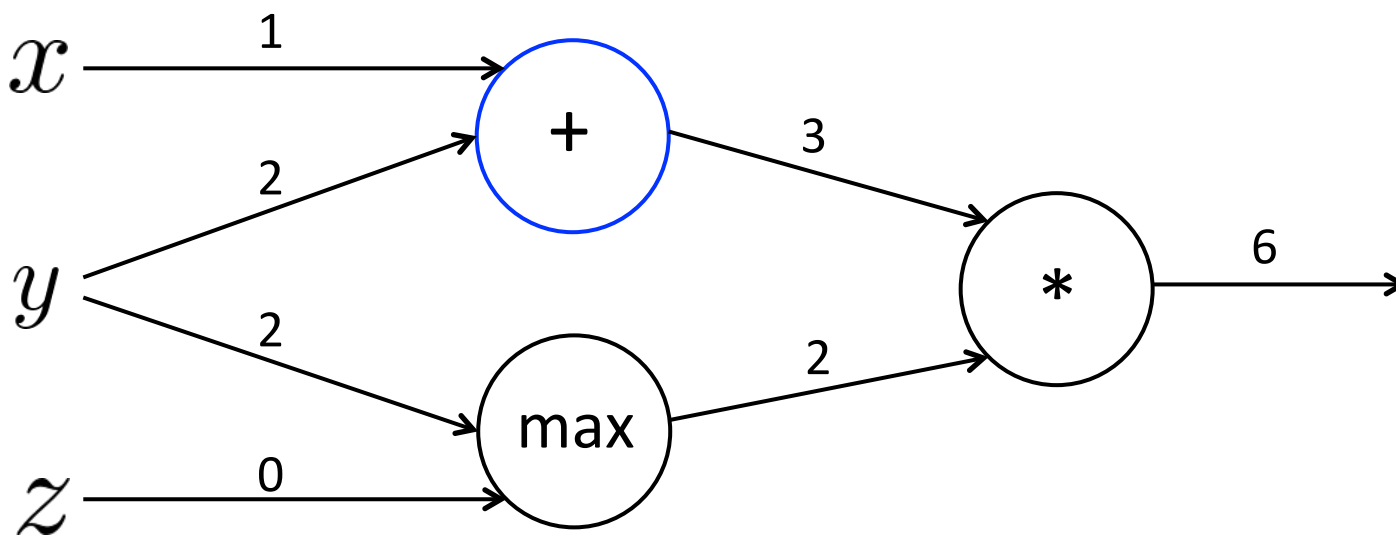
$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

Local gradients

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



An Example

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

Forward prop steps

$$a = x + y$$

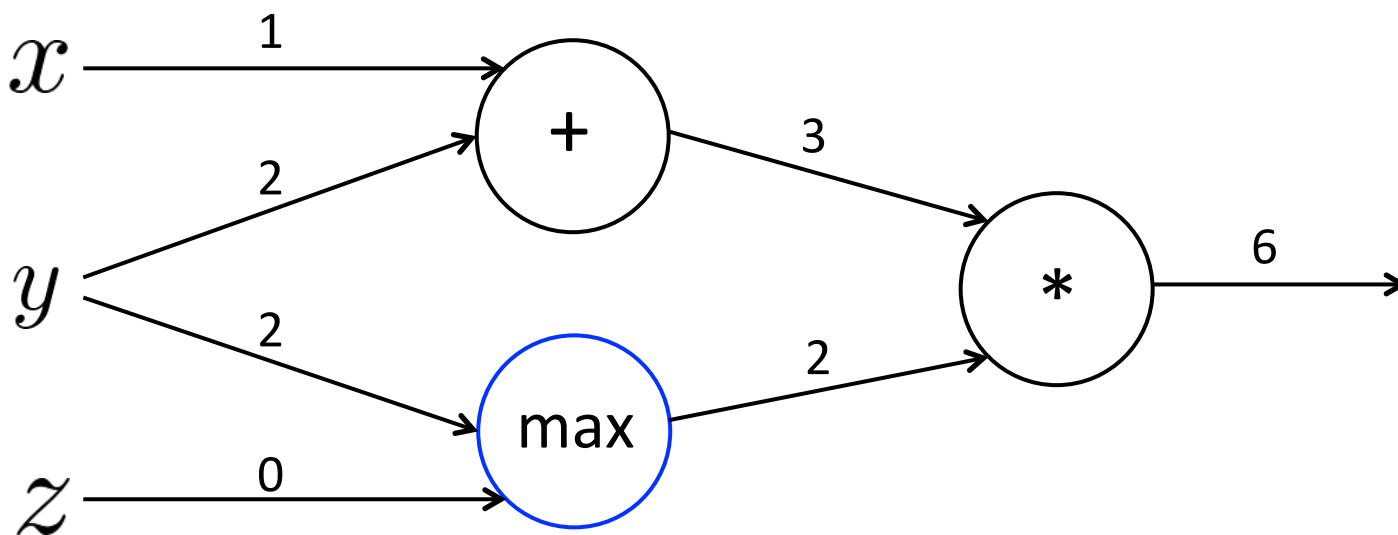
$$b = \max(y, z)$$

$$f = ab$$

Local gradients

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

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



An Example

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

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

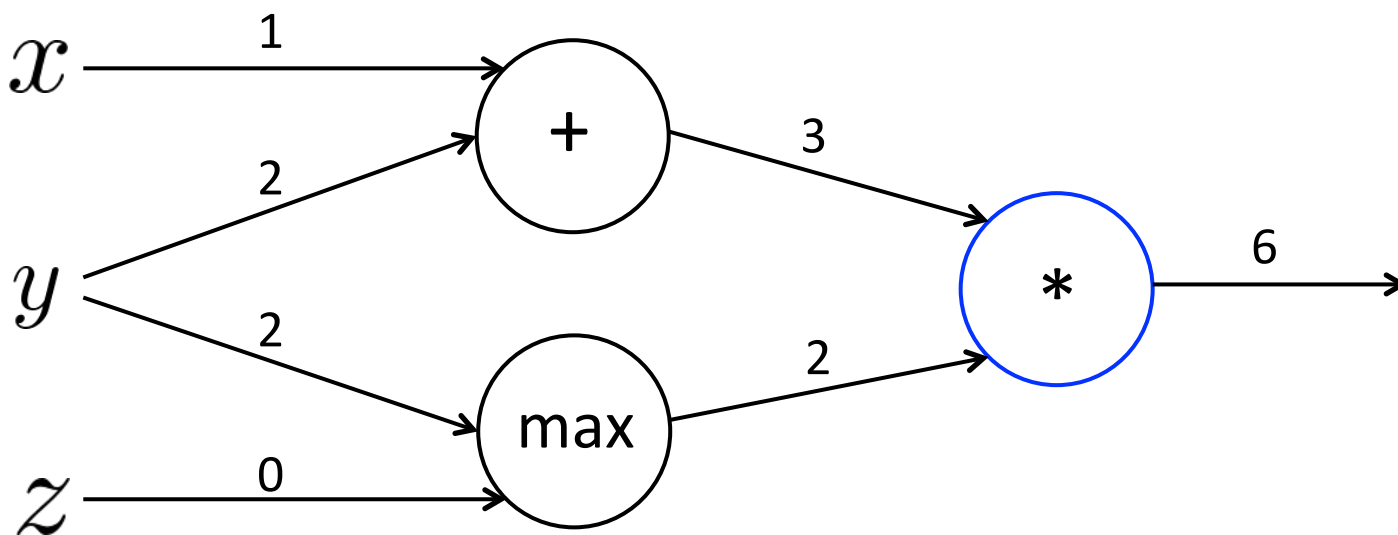
$$f = ab$$

Local gradients

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

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

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



An Example

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

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

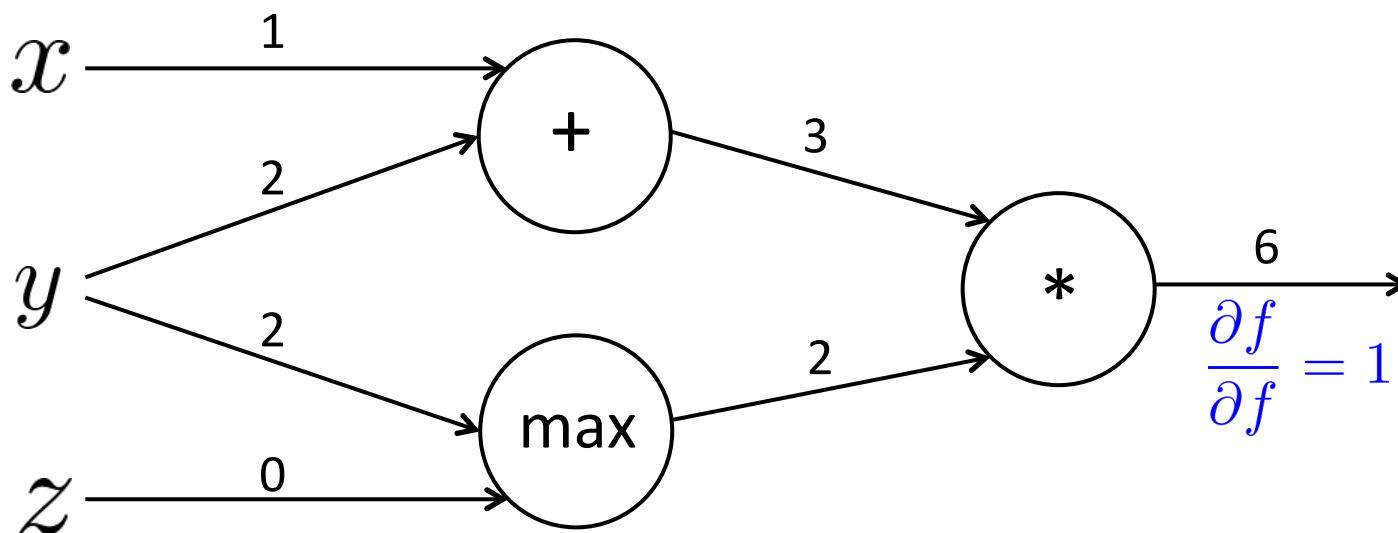
$$f = ab$$

Local gradients

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

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

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



An Example

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

$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

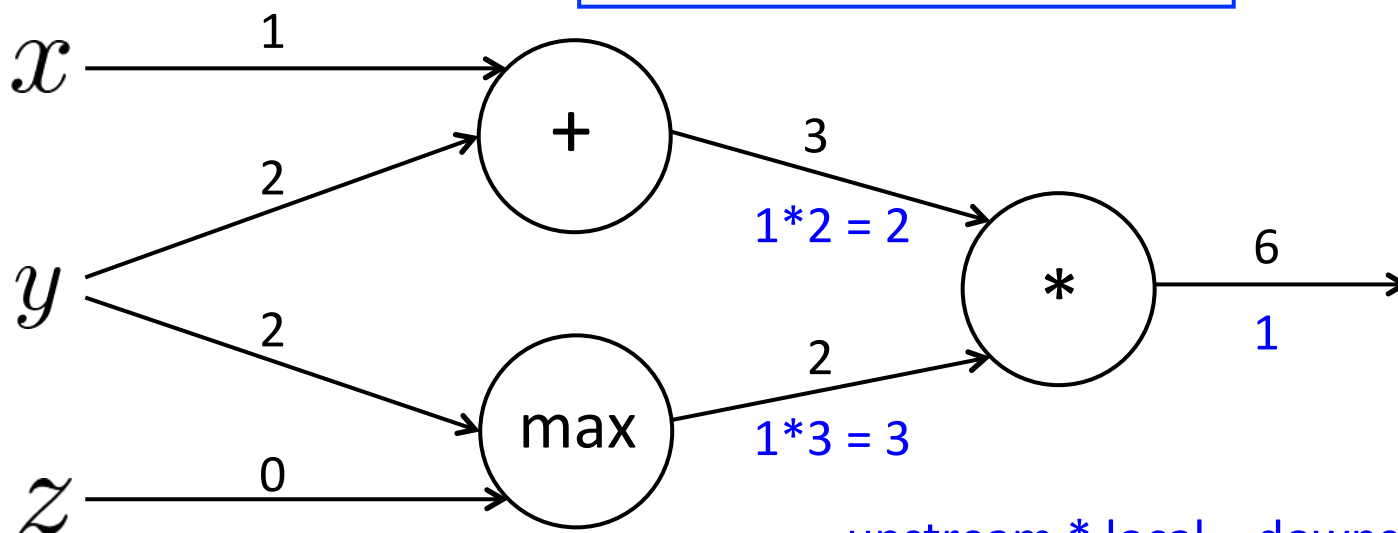
$$f = ab$$

Local gradients

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

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

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



upstream * local = downstream

An Example

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

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

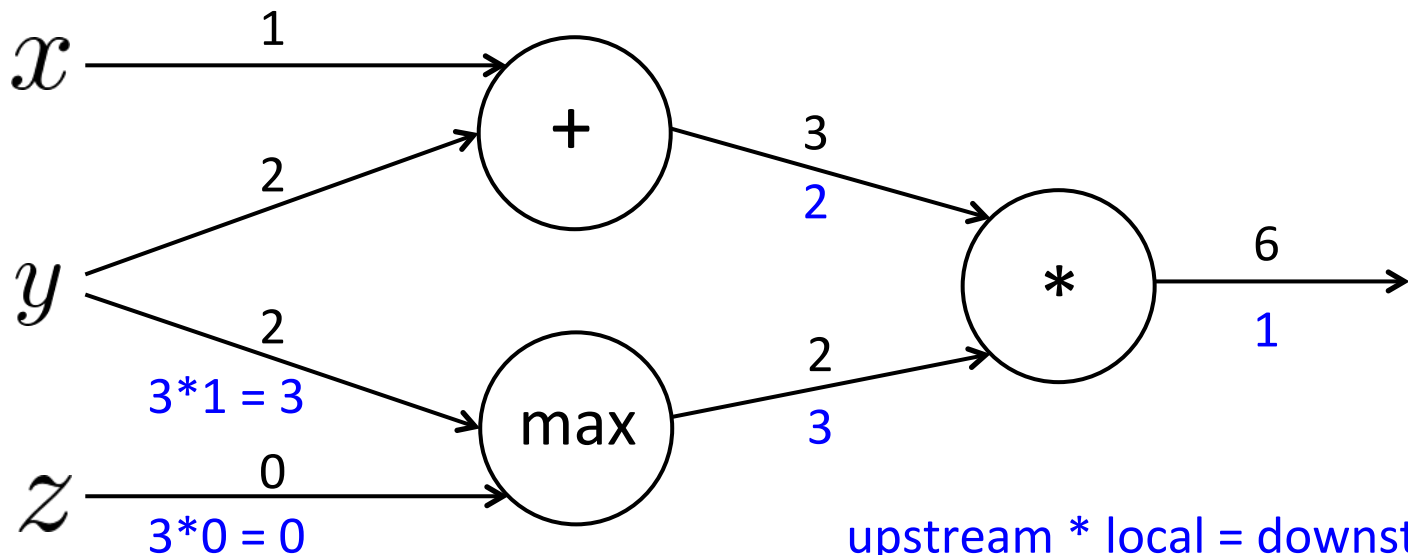
$$f = ab$$

Local gradients

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

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

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



An Example

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

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

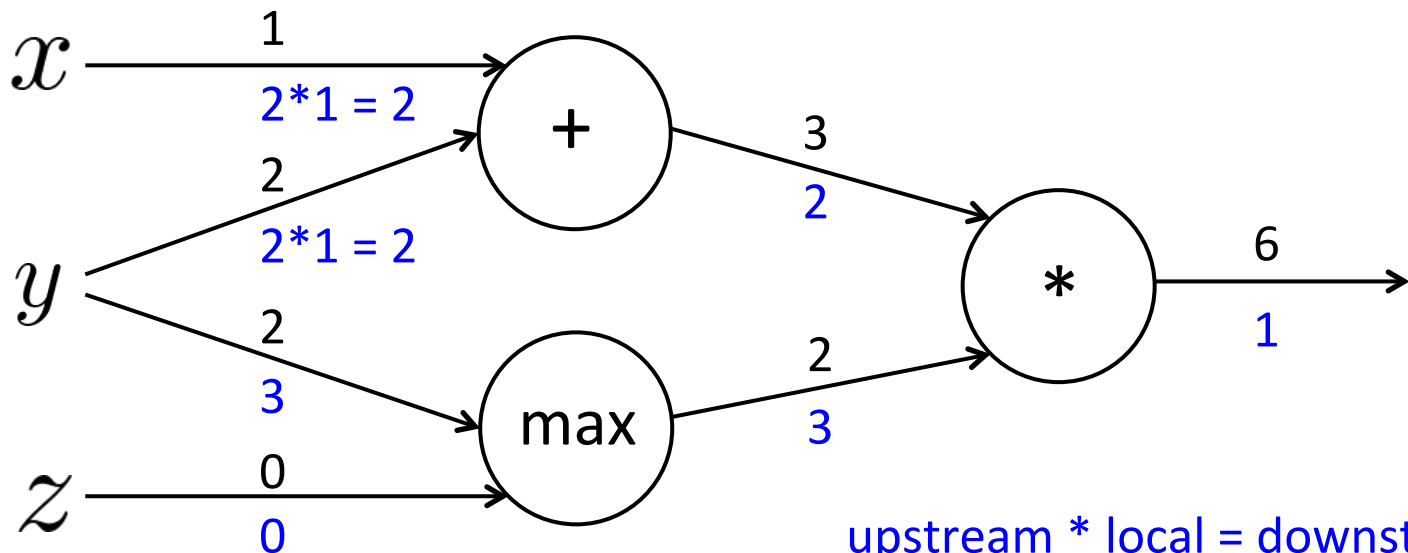
$$f = ab$$

Local gradients

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

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

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



An Example

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

$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

Local gradients

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

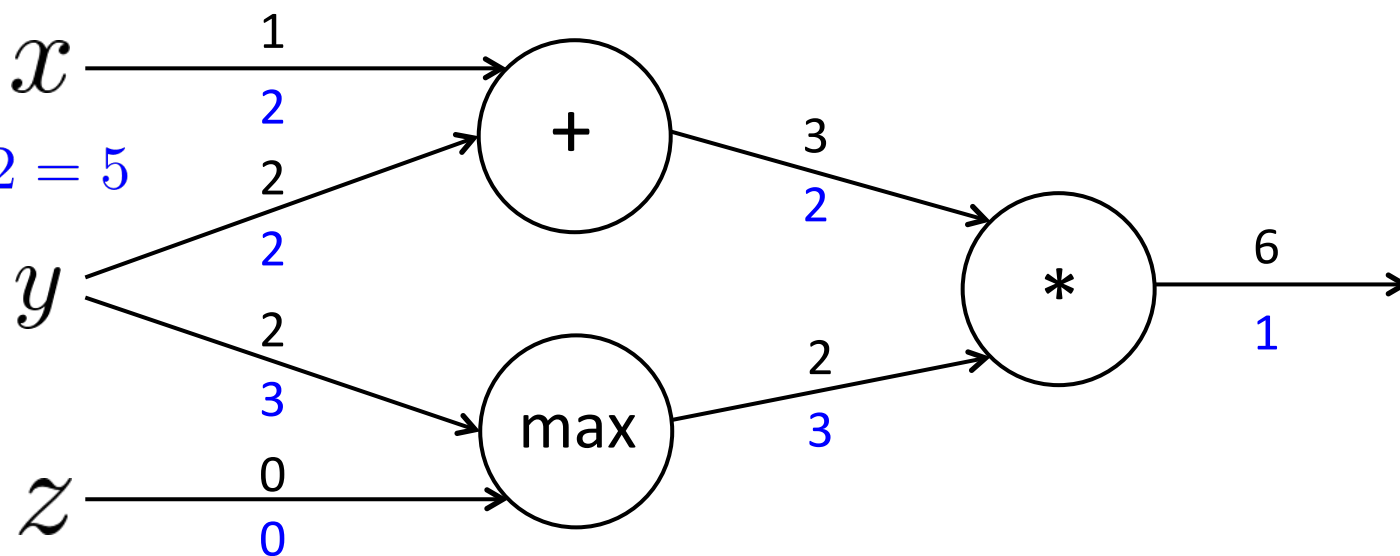
$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

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

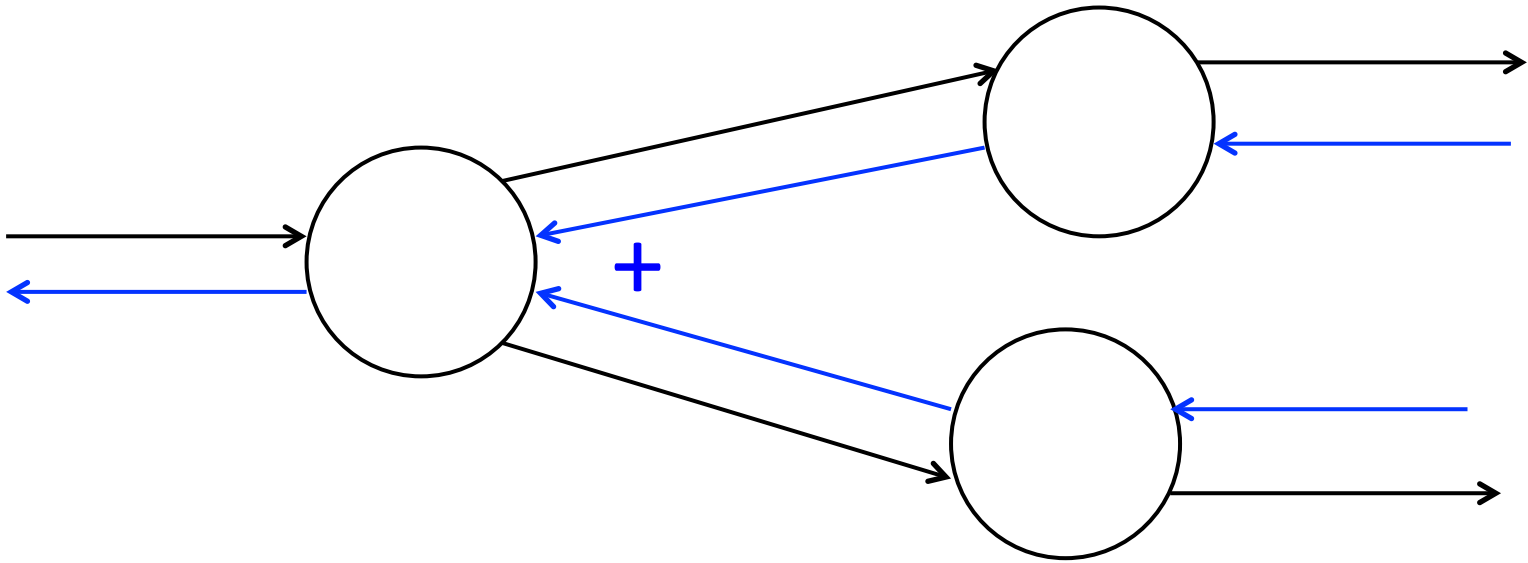
$$\frac{\partial f}{\partial x} = 2$$

$$\frac{\partial f}{\partial y} = 3 + 2 = 5$$

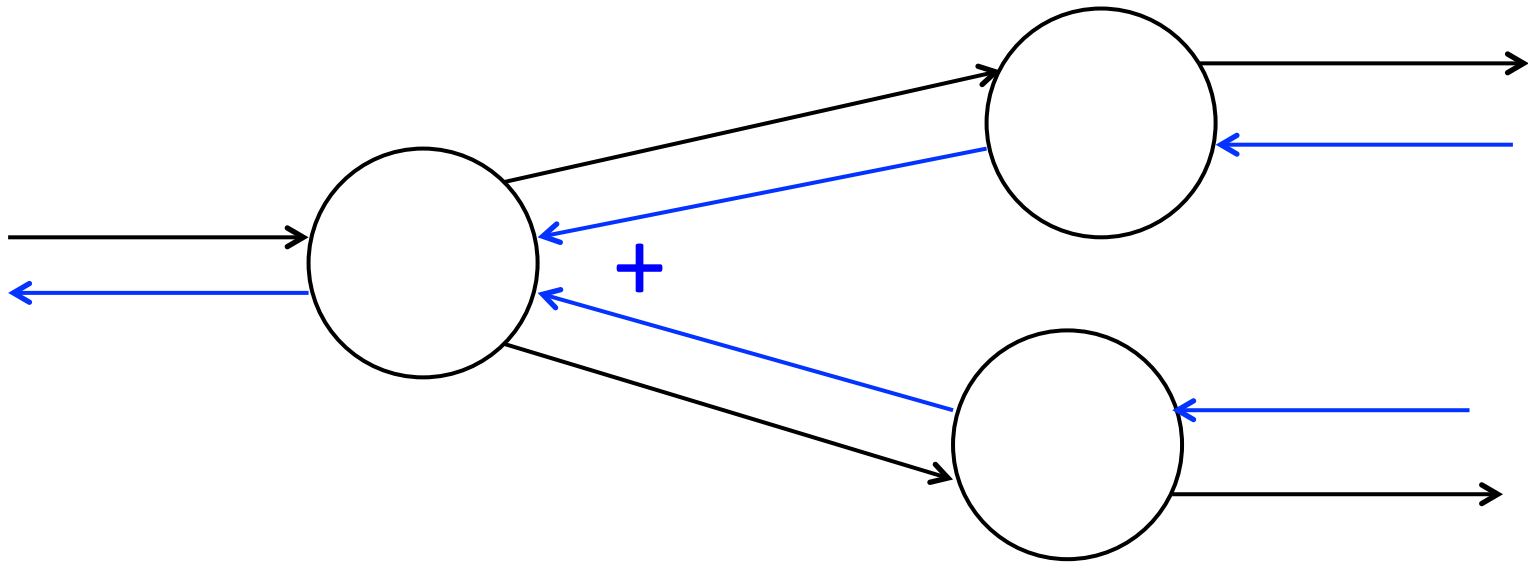
$$\frac{\partial f}{\partial z} = 0$$



Gradients add at branches



Gradients add at branches



$$a = x + y$$

$$b = \max(y, z)$$

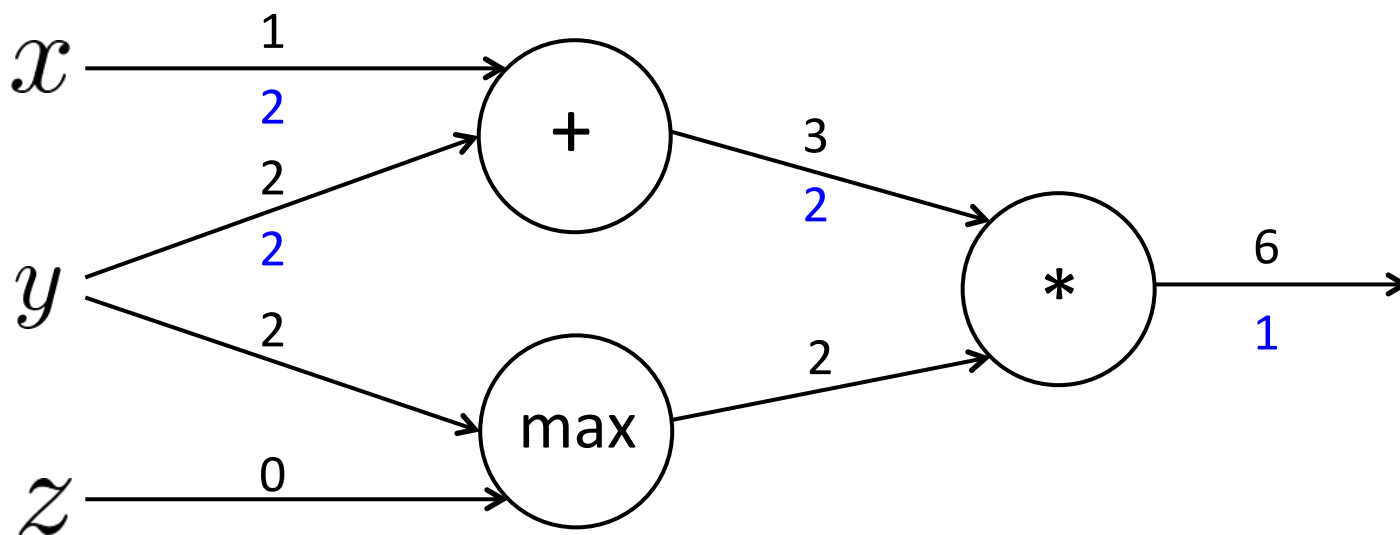
$$f = ab$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial a} \frac{\partial a}{\partial y} + \frac{\partial f}{\partial b} \frac{\partial b}{\partial y}$$

Node Intuitions

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

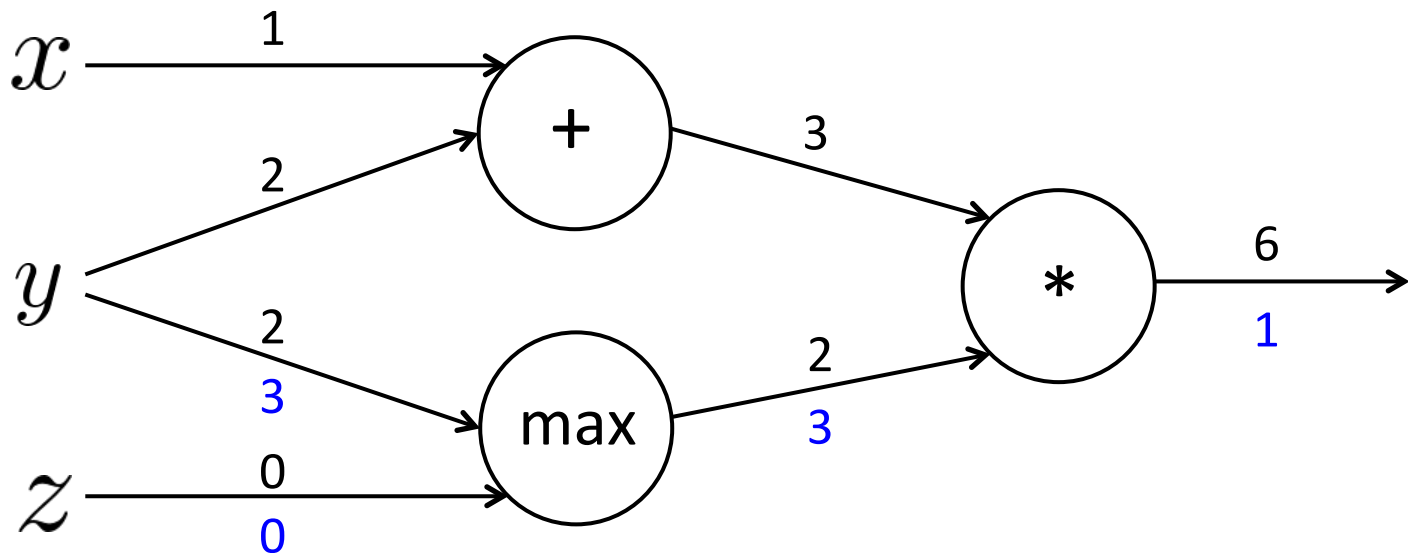
- + “distributes” the upstream gradient



Node Intuitions

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

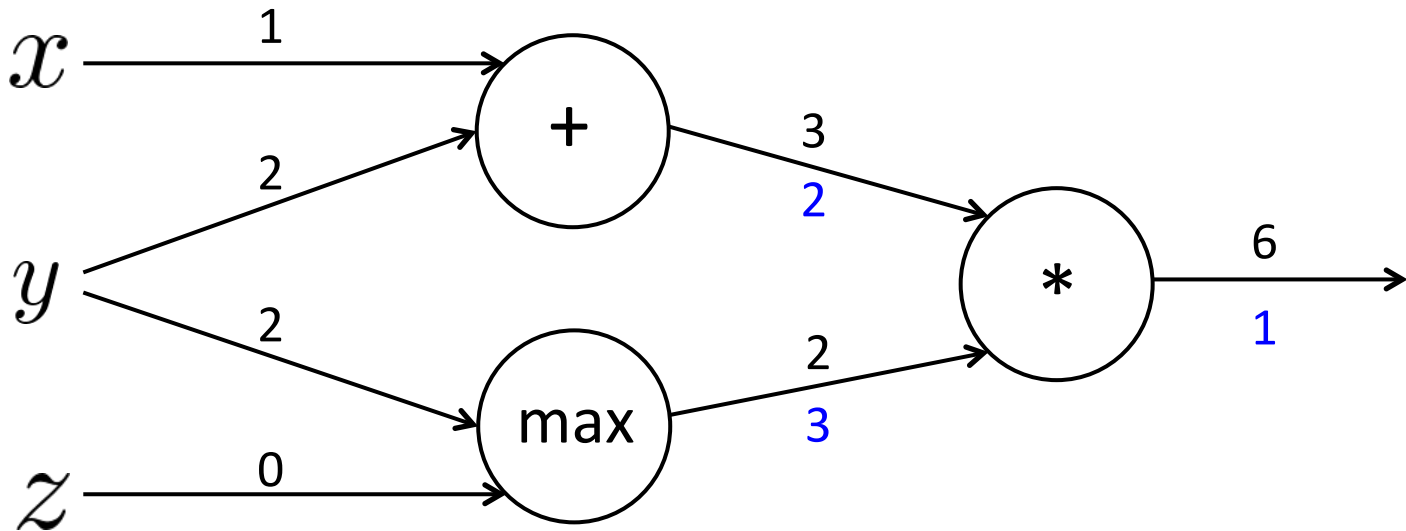
- + “distributes” the upstream gradient
- max “routes” the upstream gradient



Node Intuitions

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

- + “distributes” the upstream gradient
- max “routes” the upstream gradient
- * “switches” the upstream gradient



Efficiency: compute all gradients at once

- Incorrect way of doing backprop:

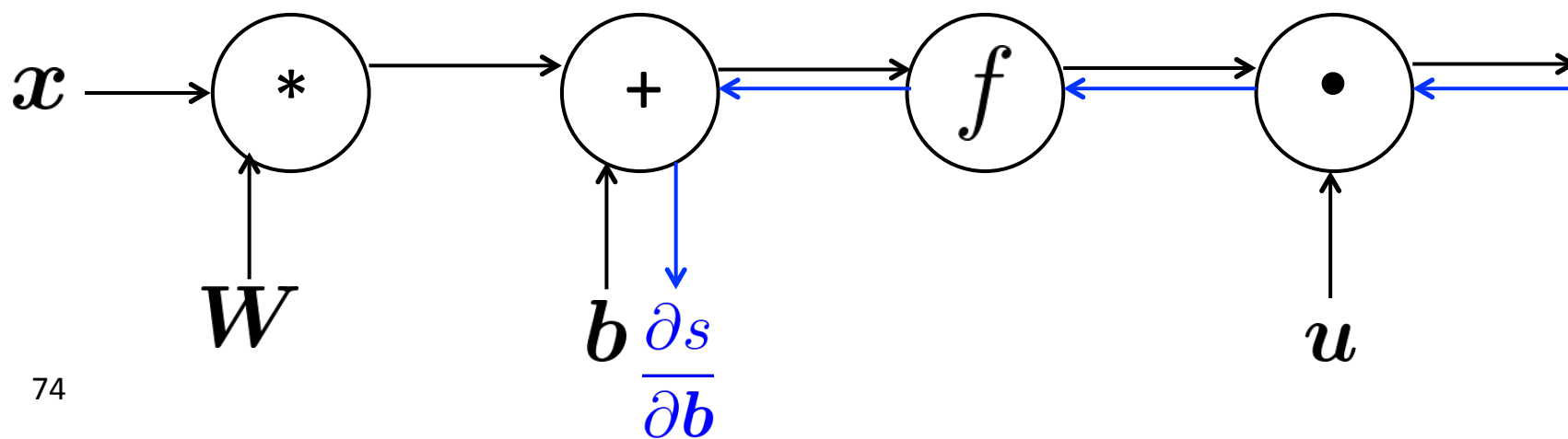
- First compute $\frac{\partial s}{\partial b}$

$$s = u^T h$$

$$h = f(z)$$

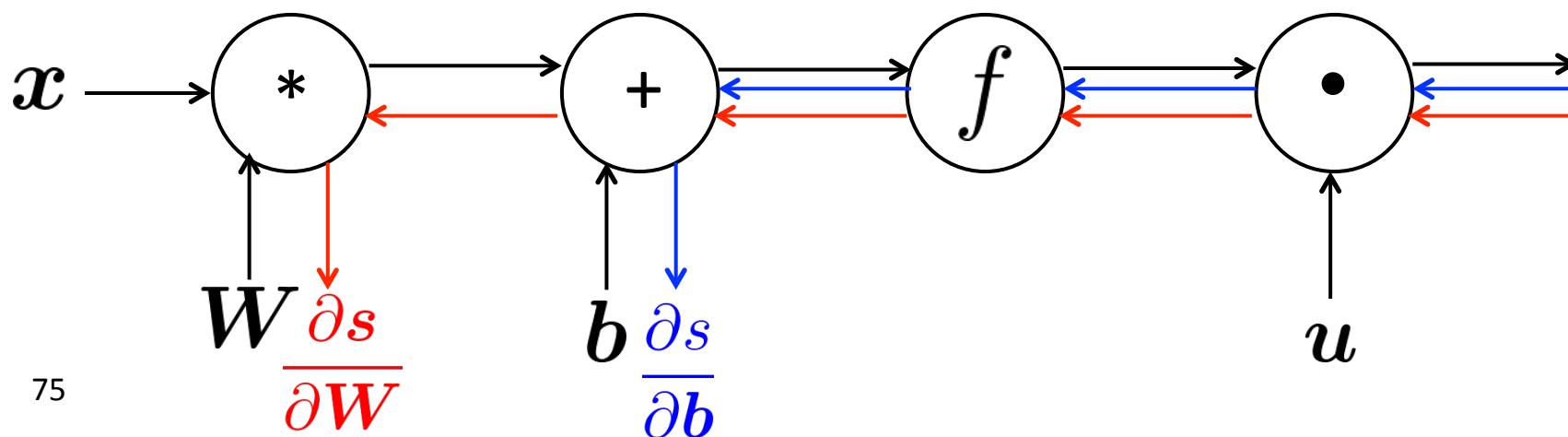
$$z = Wx + b$$

$$x \text{ (input)}$$



Efficiency: compute all gradients at once

- Incorrect way of doing backprop:
 - First compute $\frac{\partial s}{\partial b}$
 - Then independently compute $\frac{\partial s}{\partial W}$
 - Duplicated computation!
- $s = u^T h$
 $h = f(z)$
 $z = Wx + b$
 x (input)



Efficiency: compute all gradients at once

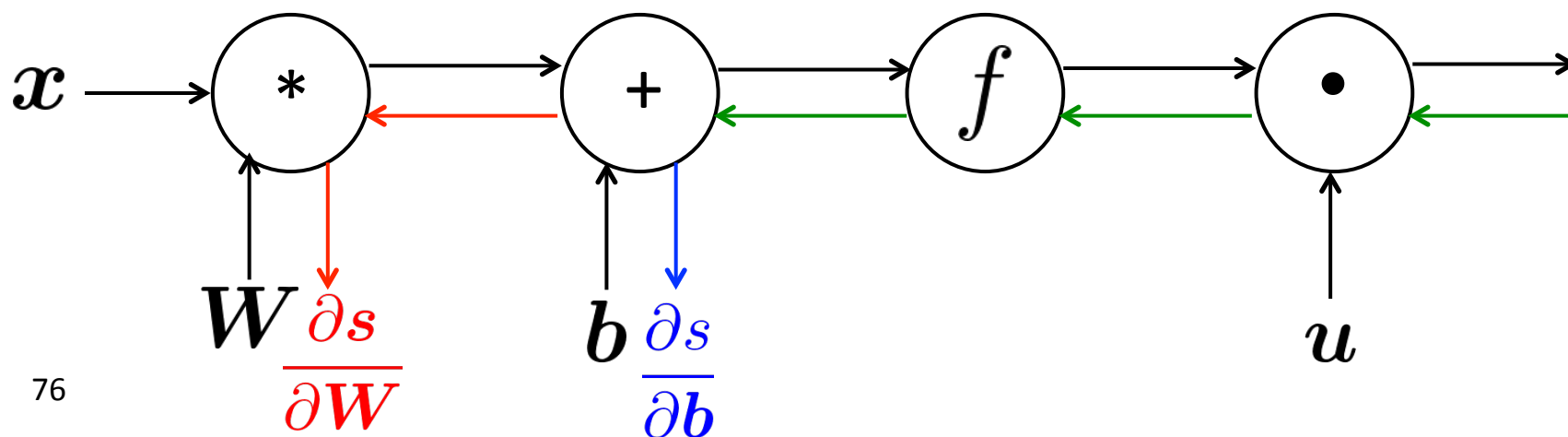
- Correct way:
 - Compute all the gradients at once
 - Analogous to using δ when we computed gradients by hand

$$s = u^T h$$

$$h = f(z)$$

$$z = \mathbf{W}x + b$$

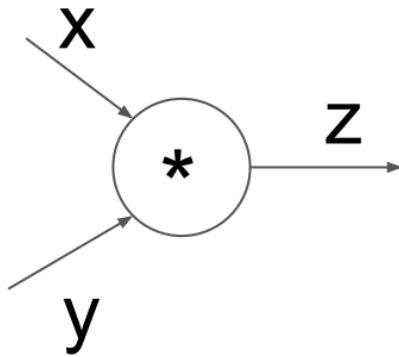
$$x \quad (\text{input})$$



Backprop Implementations

```
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
```

Implementation: forward/backward API



(x,y,z are scalars)

```
class MultiplyGate(object):
```

```
    def forward(x,y):
```

```
        z = x*y
```

```
        return z
```

```
    def backward(dz):
```

```
        # dx = ... #todo
```

```
        # dy = ... #todo
```

```
        return [dx, dy]
```

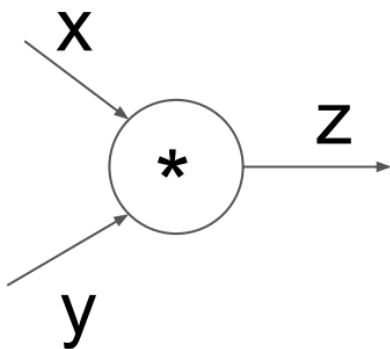
$$\frac{\partial L}{\partial z}$$

An arrow points from this box to the 'dz' parameter in the backward method definition.

$$\frac{\partial L}{\partial x}$$

An arrow points from this box to the 'dx' element in the return list of the backward method.

Implementation: forward/backward API



(x,y,z are scalars)

```
class MultiplyGate(object):  
    def forward(x,y):  
        z = x*y  
        self.x = x # must keep these around!  
        self.y = y  
        return z  
    def backward(dz):  
        dx = self.y * dz # [dz/dx * dL/dz]  
        dy = self.x * dz # [dz/dy * dL/dz]  
        return [dx, dy]
```

Alternative to backprop: Numeric Gradient

- For small h ,
$$f'(x) \approx \frac{f(x + h) - f(x - h)}{2h}$$
- Easy to implement
- But approximate and very slow:
 - Have to recompute f for every parameter of our model
- Useful for checking your implementation

Summary

- Backpropagation: recursively apply the chain rule along computational graph
 - $[\text{downstream gradient}] = [\text{upstream gradient}] \times [\text{local gradient}]$
- Forward pass: compute results of operation and save intermediate values
- Backward: apply chain rule to compute gradient

Project Types

1. Apply existing neural network model to a new task
 2. Implement a complex neural architecture(s)
 - This is what PA4 will have you do!
 3. Come up with a new model/training algorithm/etc.
 - Get 1 or 2 working first
- See project page for some inspiration

Must-haves (choose-your-own final project)

- 10,000+ labeled examples by milestone
- Feasible task
- Automatic evaluation metric
- NLP is central

Details matter!

- Split your data into train/dev/test: only look at test for final experiments
- Look at your data, collect summary statistics
- Look at your model's outputs, do error analysis
- Tuning hyperparameters is important
- Writeup quality is important
 - Look at last-year's prize winners for examples

Project Advice

- Implement simplest possible model first (e.g., average word vectors and apply logistic regression) and improve it
 - Having a baseline system is crucial
- First overfit your model to train set (get really good training set results)
 - Then regularize it so it does well on the dev set
- Start early!

