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
  - To help you choose which project option you want to do
- Final project proposal due February 8th
  - See website for details and inspiration

Overview Today:

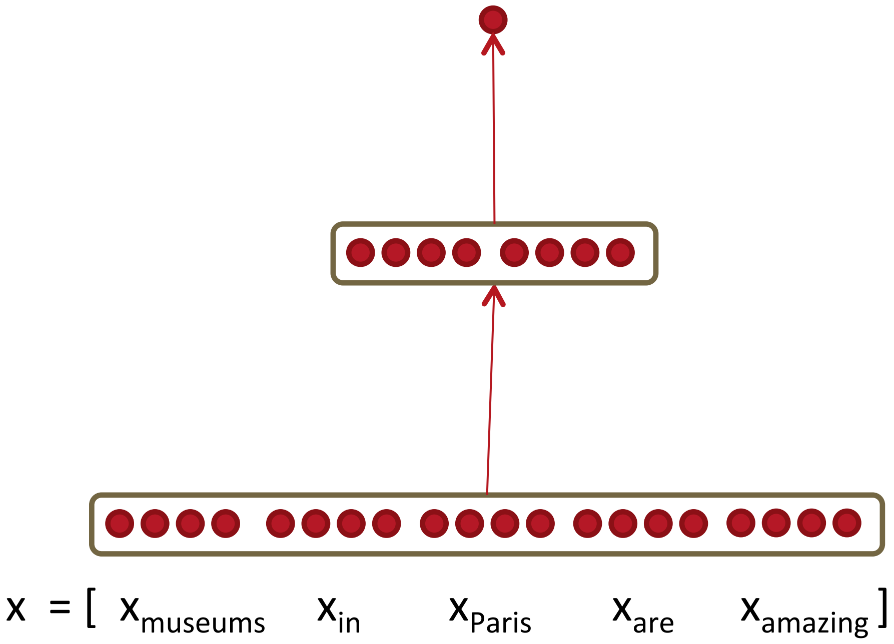
- From one-layer to multi layer neural networks!
- Fully vectorized gradient computation
- The backpropagation algorithm
- (Time permitting) Class project tips

Remember: One-layer Neural Net

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

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

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



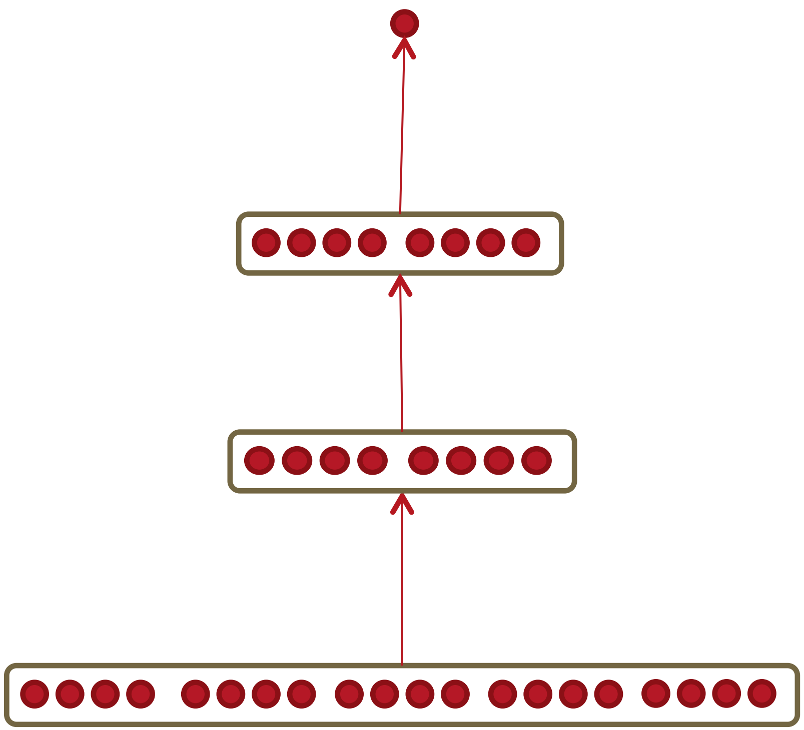
Two-layer Neural Net

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

$$\boldsymbol{h}_2 = f(\boldsymbol{W_2} \boldsymbol{h}_1 + \boldsymbol{b}_2)$$

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

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



x = [$x_{museums}$ x_{in} x_{Paris} x_{are} $x_{amazing}$]

Repeat as Needed!

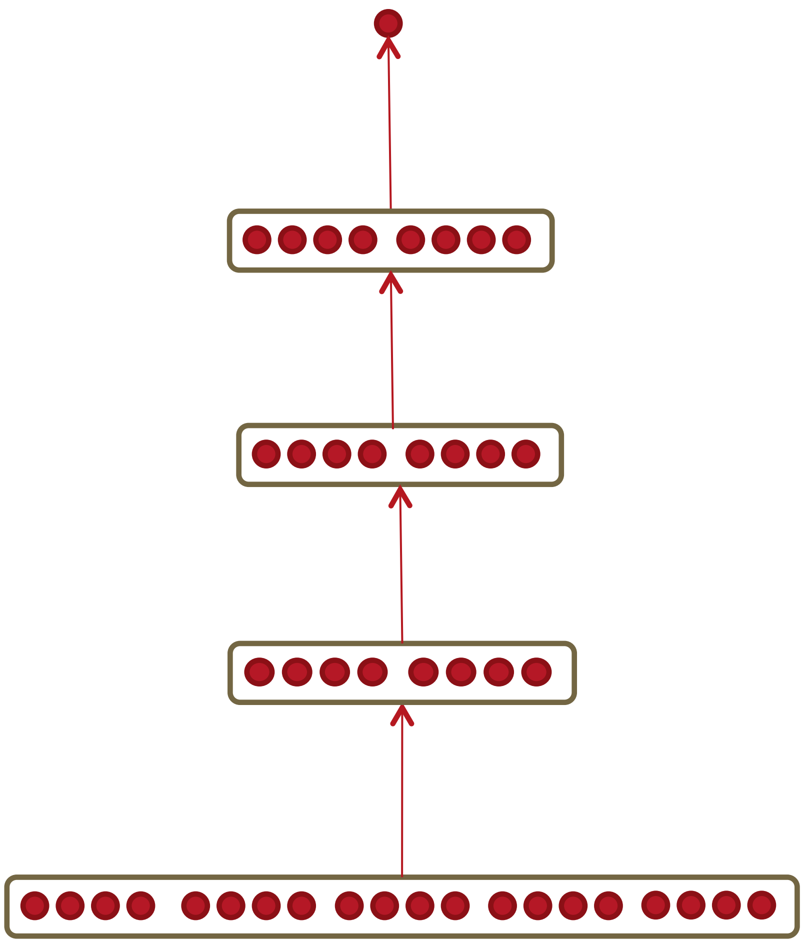
$$s = \boldsymbol{u}^T \boldsymbol{h}_3$$

$$\boldsymbol{h}_3 = f(\boldsymbol{W_3}\boldsymbol{h}_2 + \boldsymbol{b}_3)$$

$$\boldsymbol{h}_2 = f(\boldsymbol{W_2}\boldsymbol{h}_1 + \boldsymbol{b}_2)$$

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

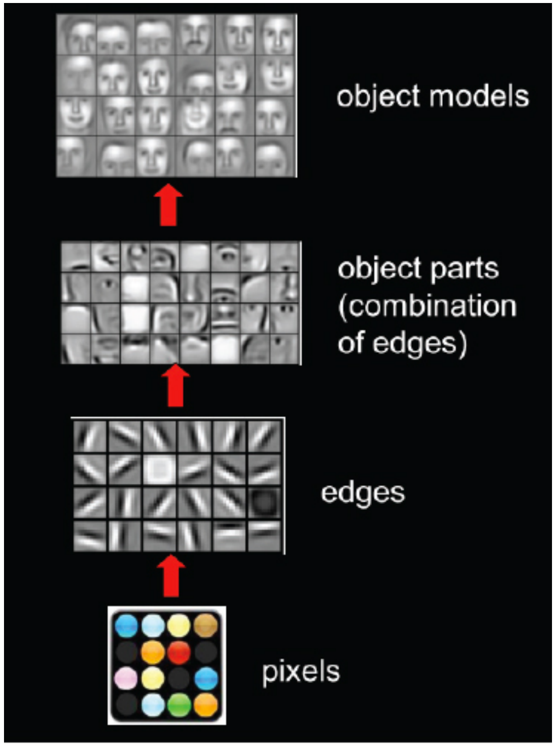
$$\boldsymbol{x} \quad \text{(input)}$$



x = [$x_{museums}$ x_{in} x_{Paris} x_{are} $x_{amazing}$]

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:

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

α = *step size* or *learning rate*

Remember: Stochastic Gradient Descent

- Update equation:

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

α = *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

$$f(\boldsymbol{x}) = f(x_1, x_2, ..., x_n)$$

- Its gradient is a vector of partial derivatives

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

Jacobian Matrix: Generalization of the Gradient

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

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

- Its Jacobian is an ***m* x *n* matrix** of partial derivatives

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

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

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

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

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

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \ldots$$

Example Jacobian: Activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

Example Jacobian: Activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

Function has *n* outputs and *n* inputs -> *n* by *n* Jacobian

Example Jacobian: Activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$

Example Jacobian: Activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$

$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

Example Jacobian: Activation Function

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$

$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$

$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

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

Other Jacobians

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

Other Jacobians

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

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

Other Jacobians

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

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

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

Other Jacobians

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

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

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

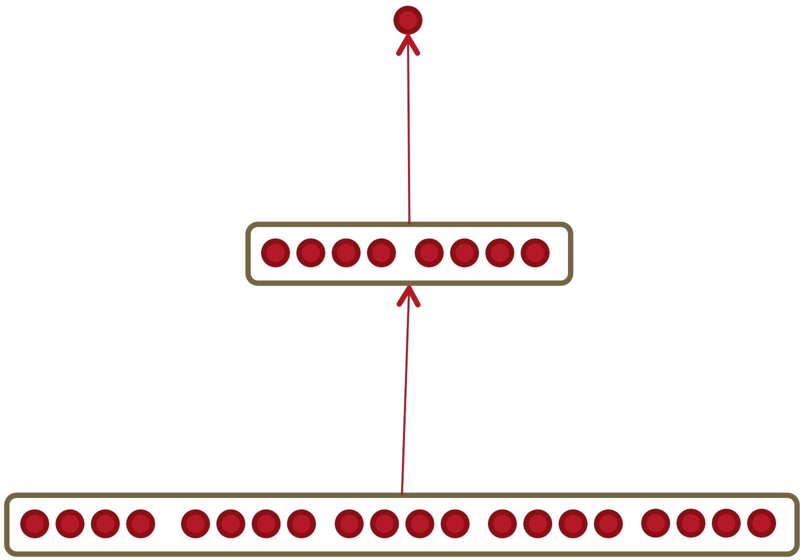
- Compute these at home for practice!
  - Check your answers with the lecture notes

Back to Neural Nets!

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

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

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



x = [$x_{museums}$ x_{in} x_{Paris} x_{are} $x_{amazing}$]

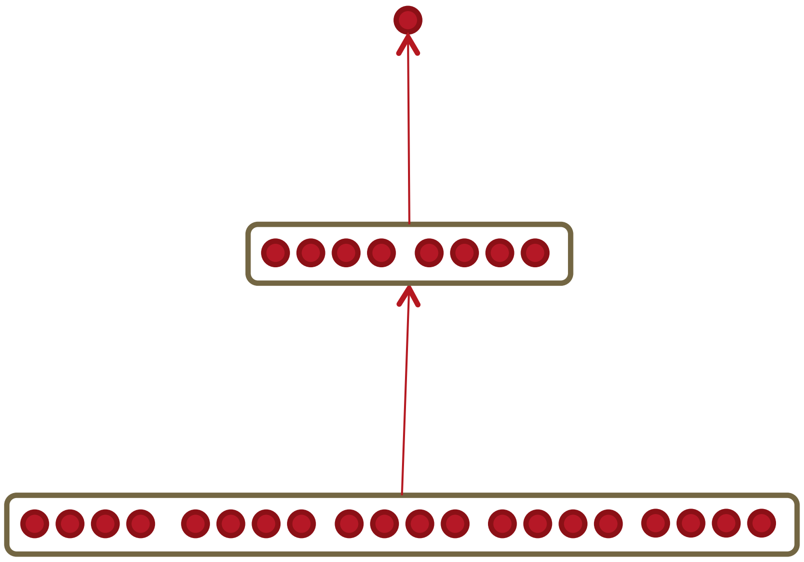
Back to Neural Nets!

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

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

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

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



x = [$x_{museums}$ x_{in} x_{Paris} x_{are} $x_{amazing}$]

1. Break up equations into simple pieces

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

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

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

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

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

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

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

2. Apply the chain rule

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

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

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

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

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

2. Apply the chain rule

$$\boxed{s = \boldsymbol{u}^T \boldsymbol{h}}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \boxed{\frac{\partial s}{\partial \boldsymbol{h}}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

2. Apply the chain rule

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

$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$

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

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

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \boxed{\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

2. Apply the chain rule

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

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

$$\boxed{\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}}$$

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

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \boxed{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}$$

3. Write out the Jacobians

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \boldsymbol{h}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{u}^T$$
$$\frac{\partial}{\partial \boldsymbol{z}}(f(\boldsymbol{z})) = \text{diag}(f'(\boldsymbol{z}))$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I}$$

3. Write out the Jacobians

$$\boxed{s = \boldsymbol{u}^T \boldsymbol{h}}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$= \boldsymbol{u}^T$$

Useful Jacobians from previous slide

$$\boxed{\frac{\partial}{\partial \boldsymbol{h}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{u}^T}$$
$$\frac{\partial}{\partial \boldsymbol{z}}(f(\boldsymbol{z})) = \text{diag}(f'(\boldsymbol{z}))$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I}$$

3. Write out the Jacobians

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

$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$

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

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

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$= \boldsymbol{u}^T \text{diag}(\text{f'}(\boldsymbol{z}))$$

Useful Jacobians from previous slide

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

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

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

3. Write out the Jacobians

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

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

$$\boxed{\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}}$$

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

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$= \boldsymbol{u}^T \text{diag}(\text{f'}(\boldsymbol{z}))\boldsymbol{I}$$

Useful Jacobians from previous slide

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

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

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

3. Write out the Jacobians

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$= \boldsymbol{u}^T \text{diag}(\text{f'}(\boldsymbol{z}))\boldsymbol{I}$$

$$= \boldsymbol{u}^T \circ f'(\boldsymbol{z})$$

Useful Jacobians from previous slide

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

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

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

Re-using Computation

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

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

Re-using Computation

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

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

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

The same! Let’s avoid duplicated computation…

Re-using Computation

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

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

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

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

Derivative with respect to Matrix

- What does $\frac{\partial s}{\partial \boldsymbol{W}}$ look like? $\boldsymbol{W} \in \mathbb{R}^{n \times m}$
- 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 $\dfrac{\partial s}{\partial \boldsymbol{W}}$ look like? $\quad \boldsymbol{W} \in \mathbb{R}^{n \times m}$
- 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 $\dfrac{\partial s}{\partial \boldsymbol{W}}$ is *n* by *m*: $\begin{bmatrix} \dfrac{\partial s}{\partial W_{11}} & \cdots & \dfrac{\partial s}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial s}{\partial W_{n1}} & \cdots & \dfrac{\partial s}{\partial W_{nm}} \end{bmatrix}$

Derivative with respect to Matrix

- Remember $\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$
  - $\boldsymbol{\delta}$ is going to be in our answer
  - The other term should be $\boldsymbol{x}$ because $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$

- It turns out $\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \boldsymbol{x}^T$

Why the Transposes?

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \quad \boldsymbol{x}^T$$

$$[n \times m] \quad [n \times 1][1 \times m]$$

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

Why the Transposes?

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

- 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?

- $\dfrac{\partial s}{\partial \boldsymbol{b}} = \boldsymbol{u}^T \circ f'(\boldsymbol{z})$ is a row vector
  - But convention says our gradient should be a column vector because $\boldsymbol{b}$ 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 $\frac{\partial s}{\partial \boldsymbol{b}}$ to make the derivative a column vector, resulting in $\boldsymbol{\delta}^T$
- 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 of the score, but in PA1 its of the loss
- Don't forget to replace *f'* with the actual derivative
- PA1 uses $\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$ for the linear transformation: gradients are different!

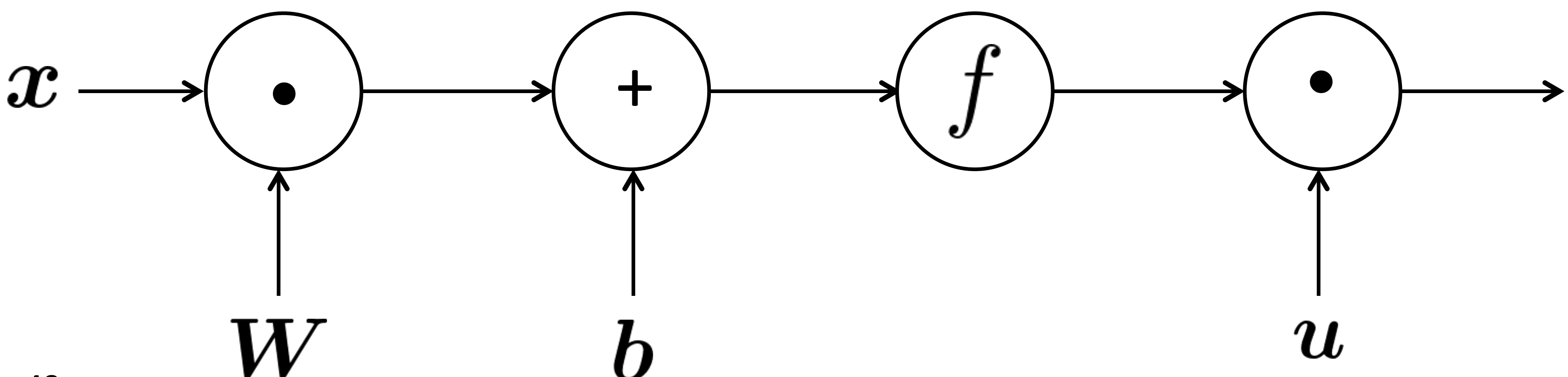
Backpropagation

- 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

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$



Computational Graphs

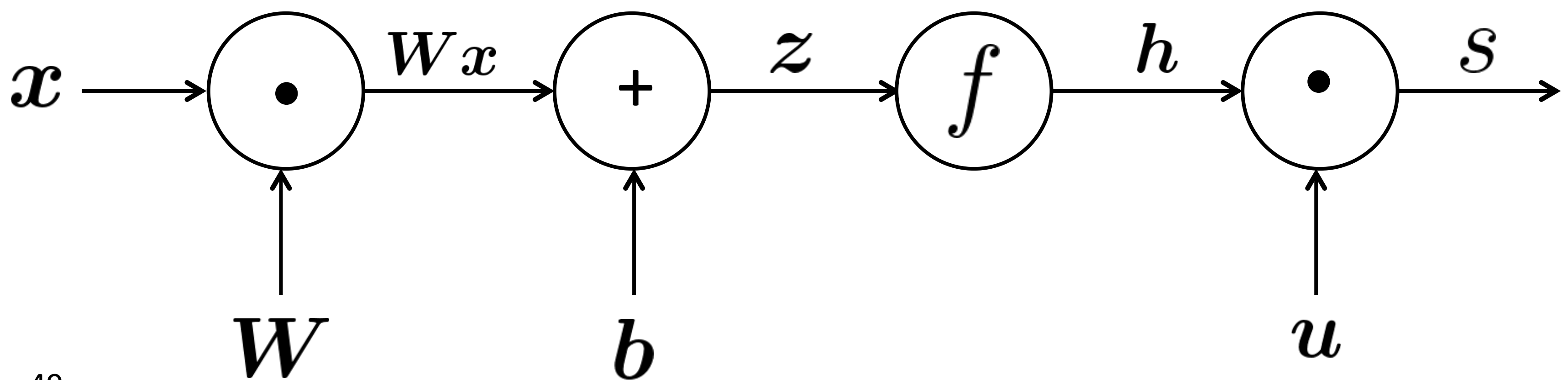
- Representing our neural net equations as a graph
  - Source nodes: inputs
  - Interior nodes: operations
  - Edges pass along result of the operation

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

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

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

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



Computational Graphs

- Representing our neural net equations as a graph

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

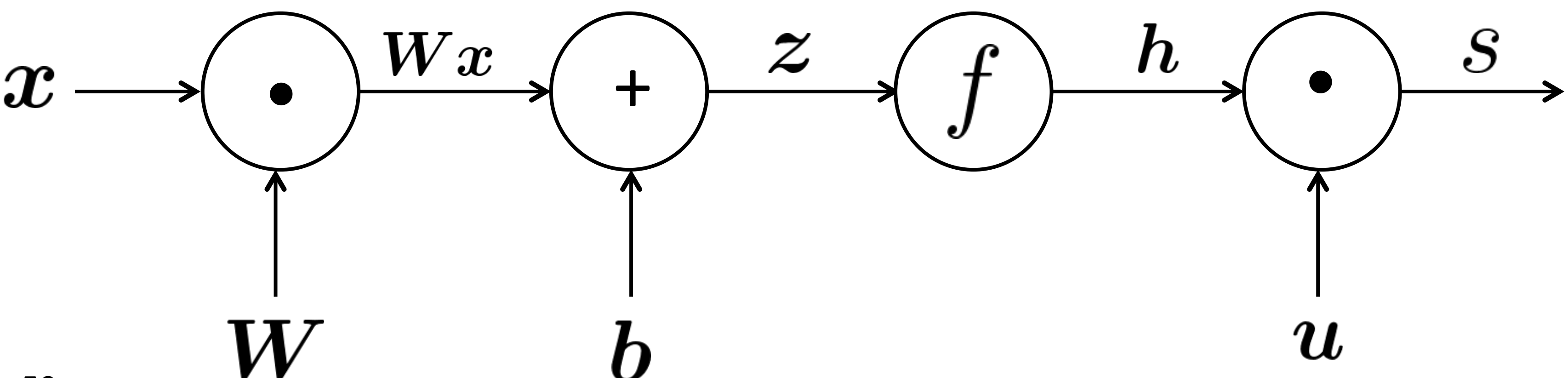
$$\boldsymbol{h} = f(\boldsymbol{z})$$

…$x + \boldsymbol{b}$

…ut)

"Forward Propagation"

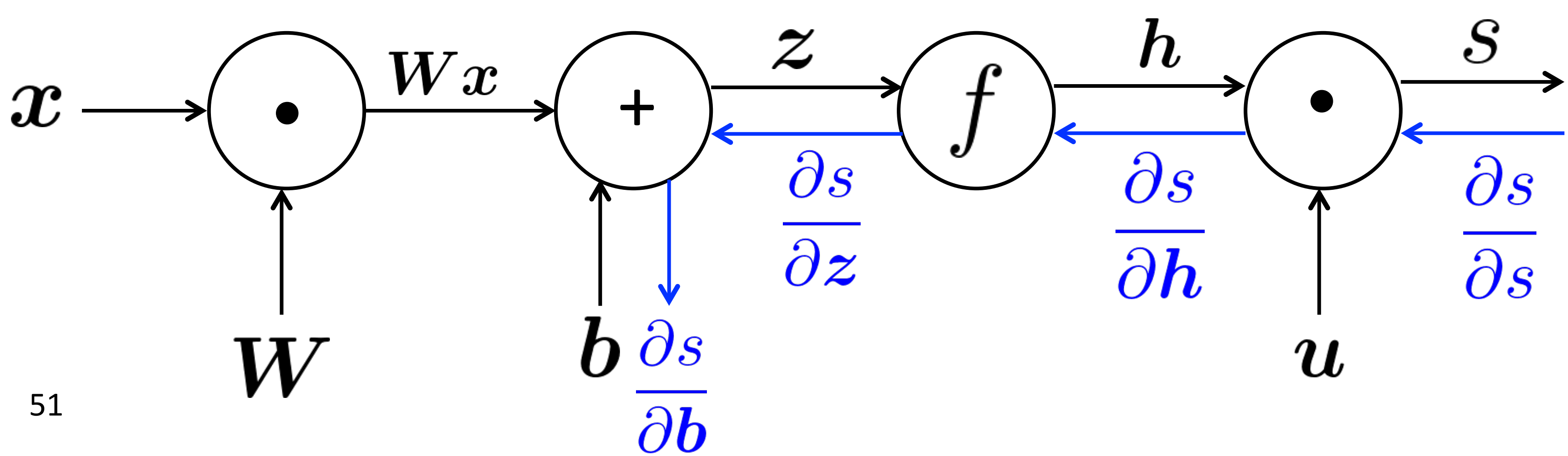
operation



Backpropagation

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

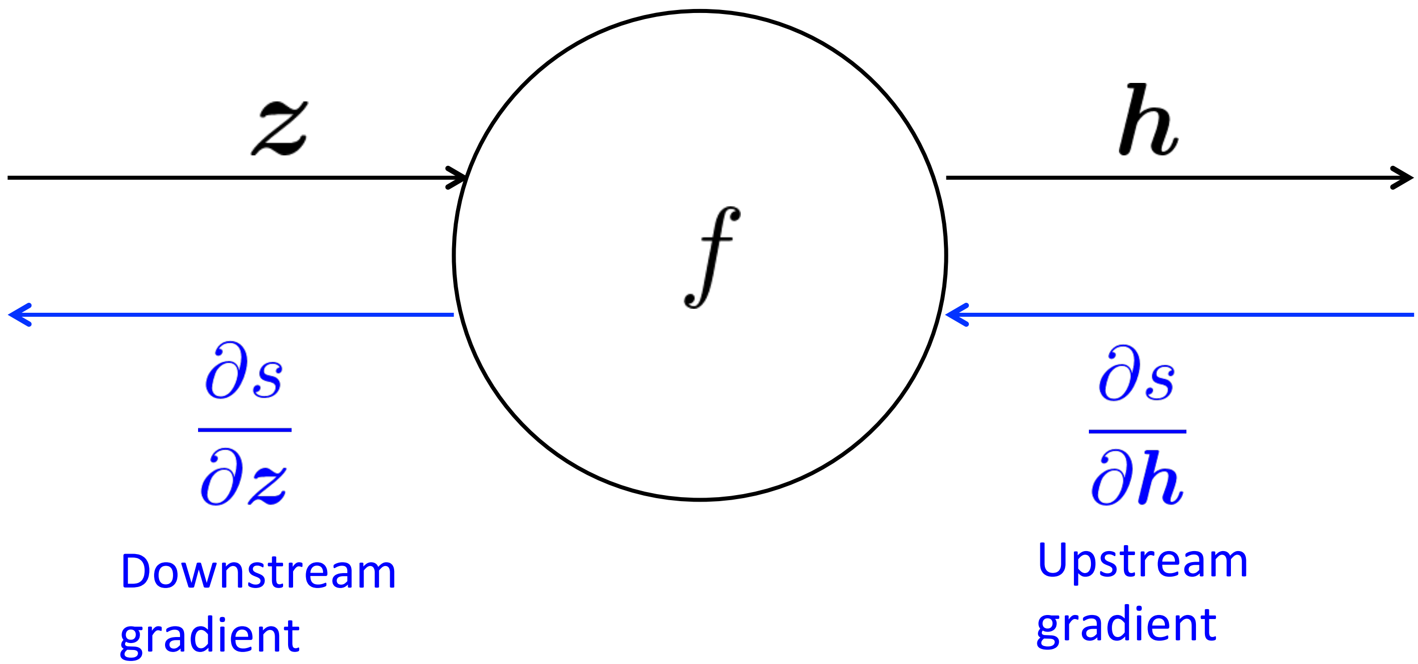
$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$



Backpropagation: Single Node

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

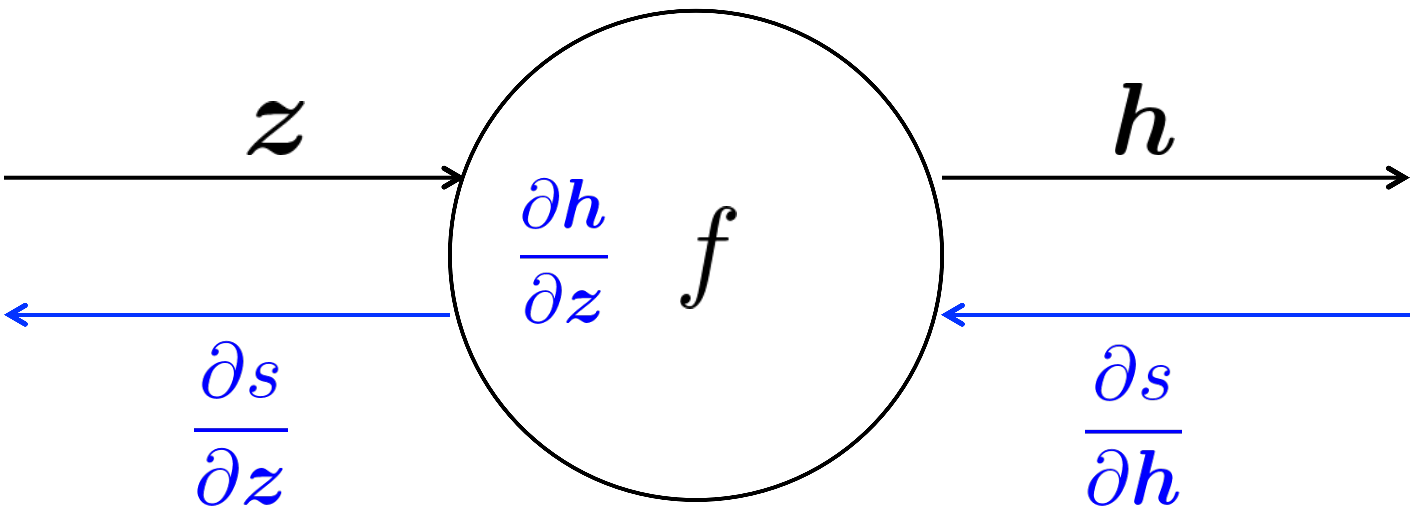
$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$



Backpropagation: Single Node

- Each node has a **local gradient**
  - The gradient of its output with respect to its input

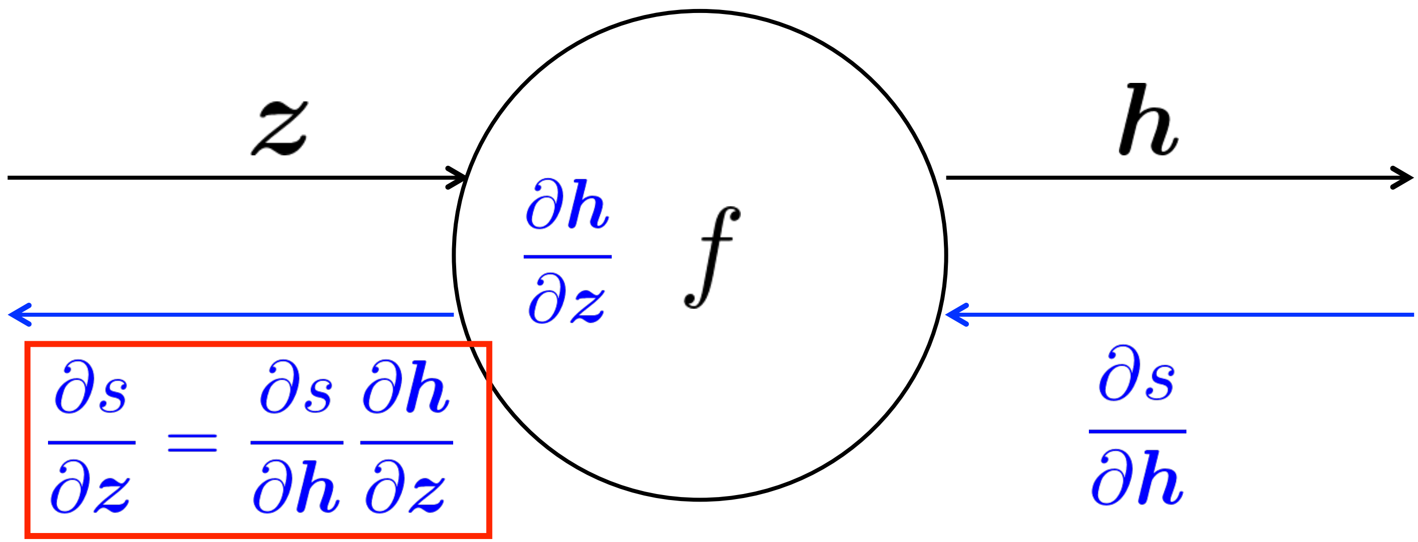
$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$



Backpropagation: Single Node

- Each node has a **local gradient**
  - The gradient of its output with respect to its input

$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$

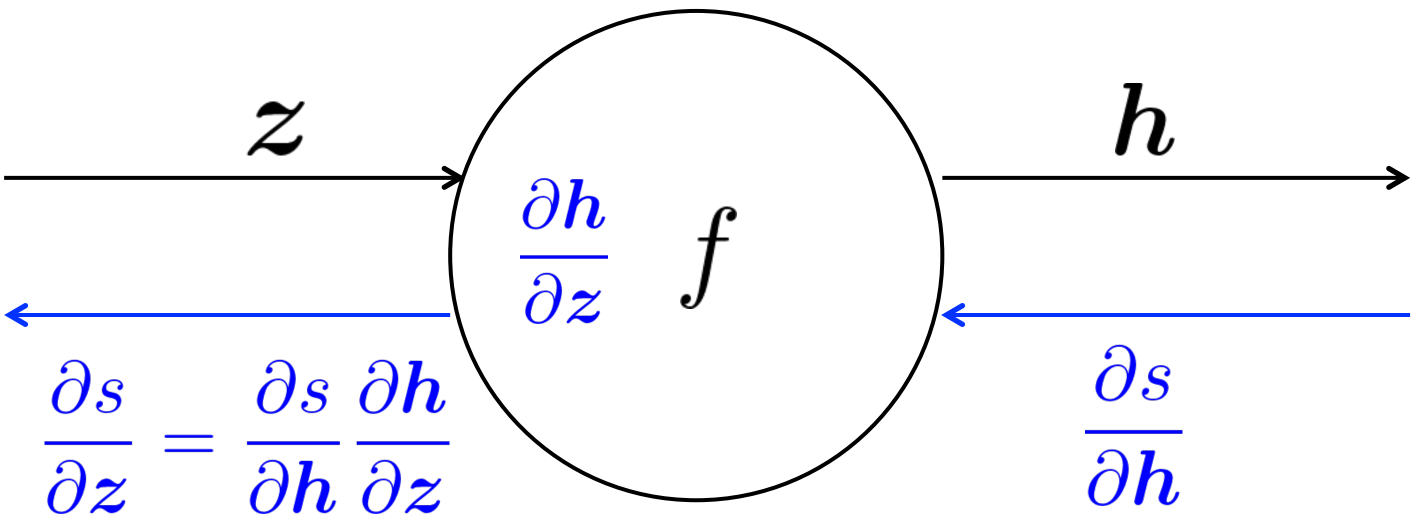


Backpropagation: Single Node

- Each node has a **local gradient**
  - The gradient of its output with respect to its input

$$\boxed{\boldsymbol{h} = f(\boldsymbol{z})}$$

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



Downstream gradient

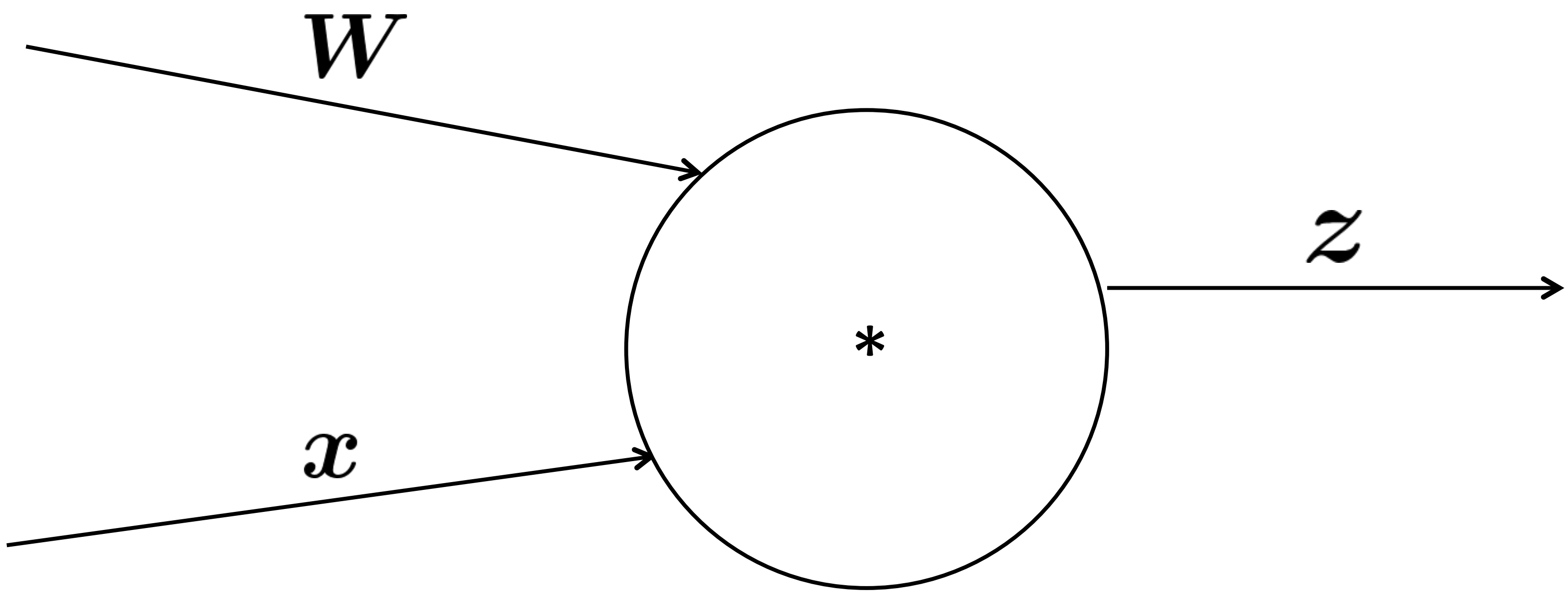
Local gradient

Upstream gradient

Backpropagation: Single Node

- What about nodes with multiple inputs?

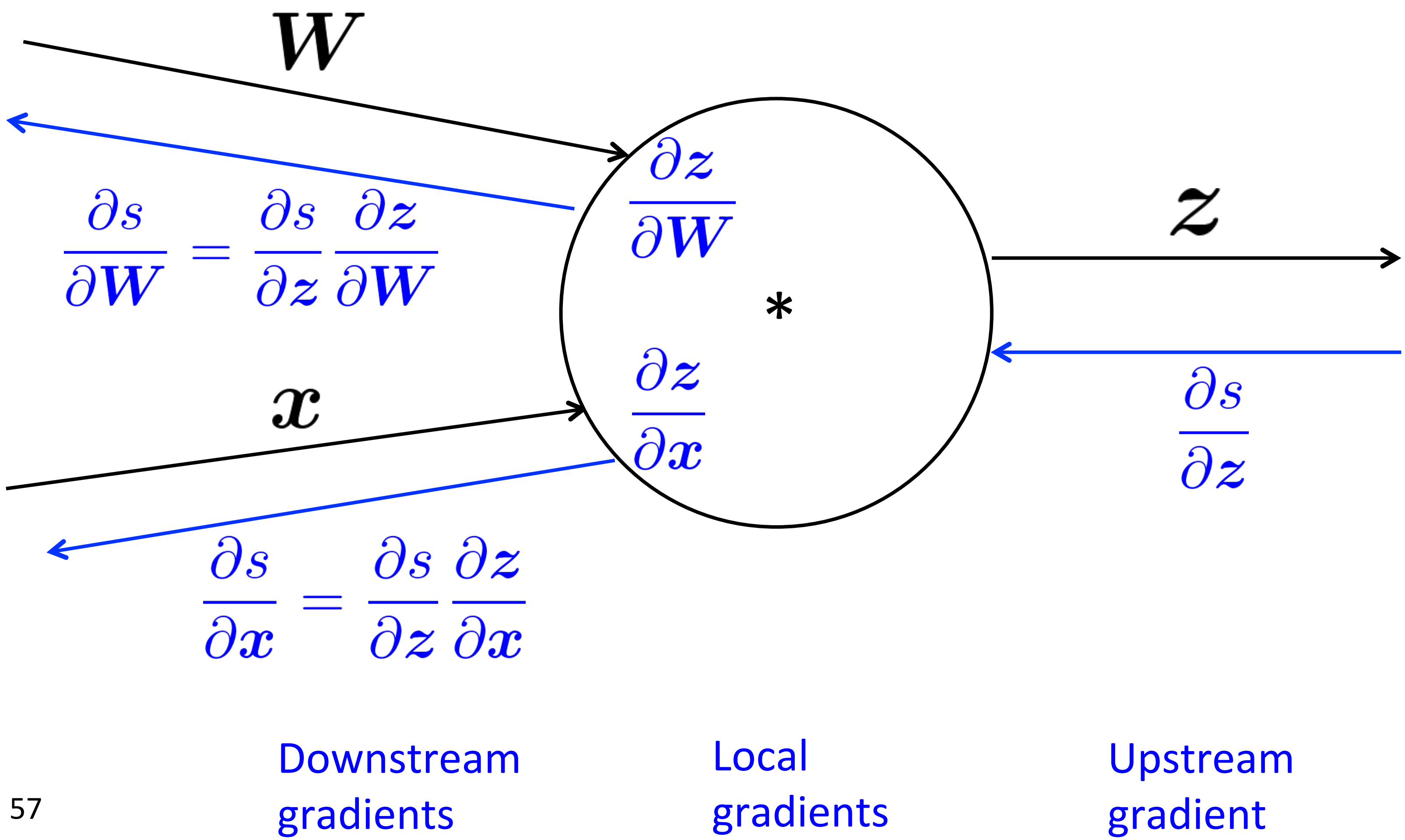
$$\boxed{\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}}$$



Backpropagation: Single Node

- Multiple inputs -> multiple local gradients

$$\boxed{\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}}$$



An Example

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

An Example

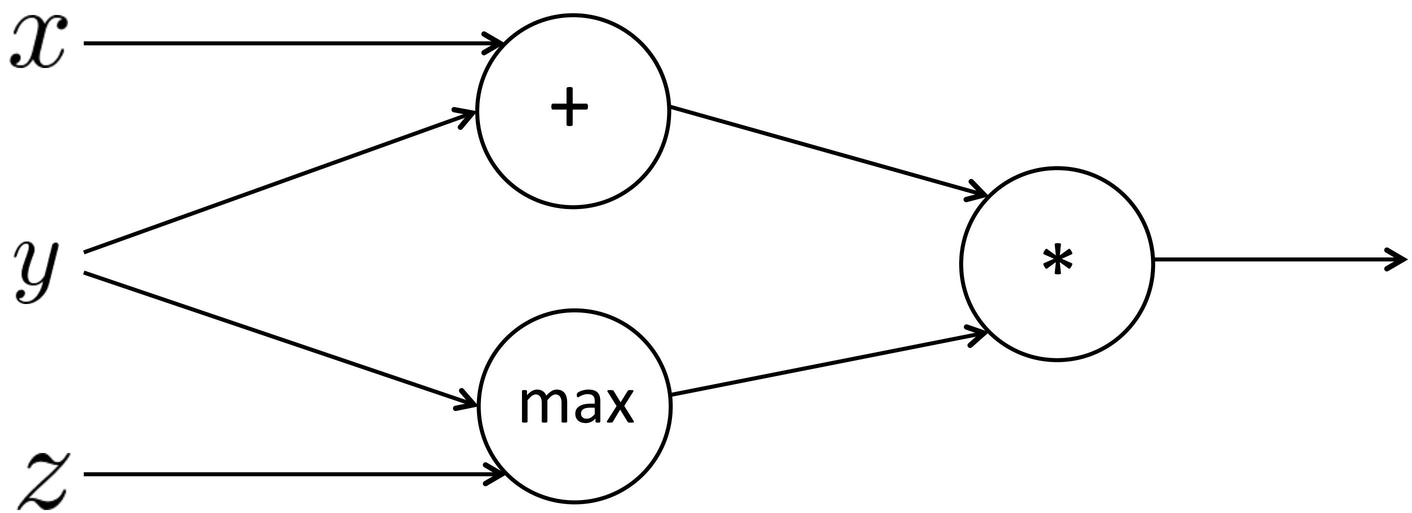
$$f(x, y, z) = (x + y)\max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

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

$$f = ab$$



An Example

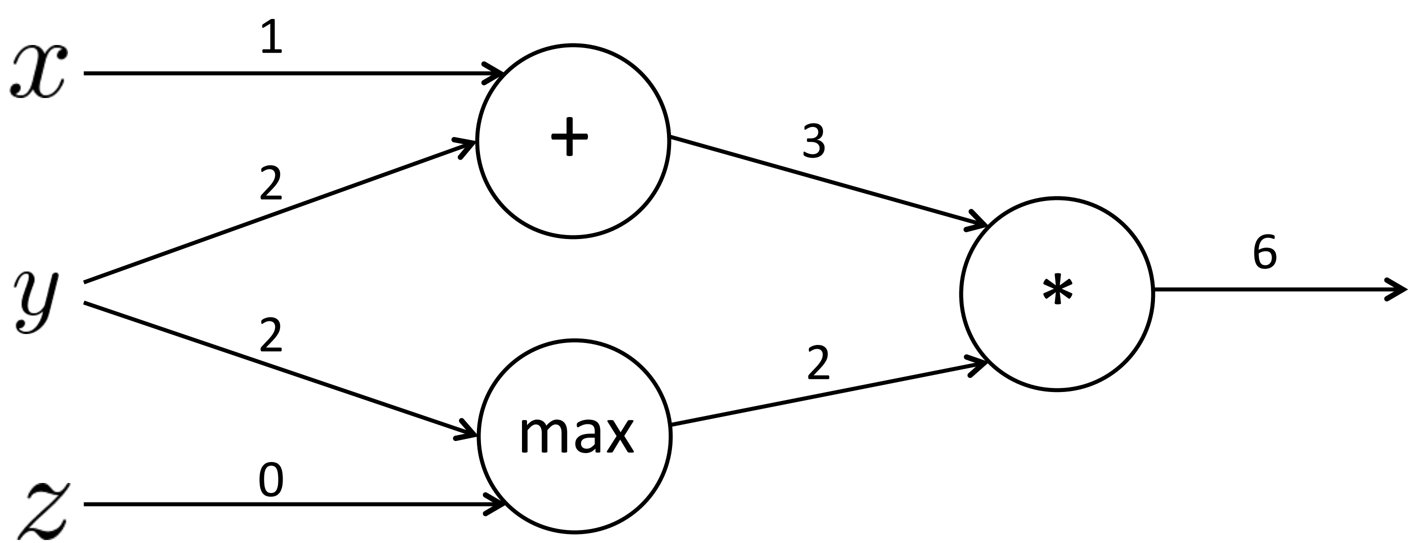
$$f(x, y, z) = (x + y)\max(y, z)$$
$$x = 1, y = 2, z = 0$$

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

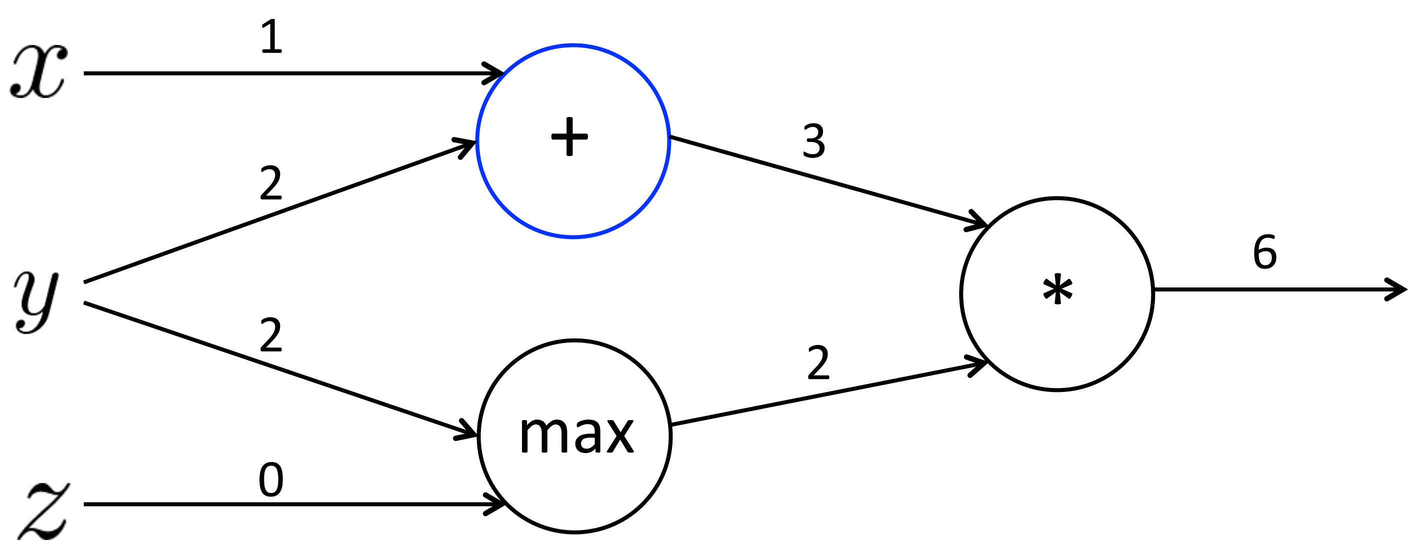
$$a = x + y$$

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

$$f = ab$$

Local gradients

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



An Example

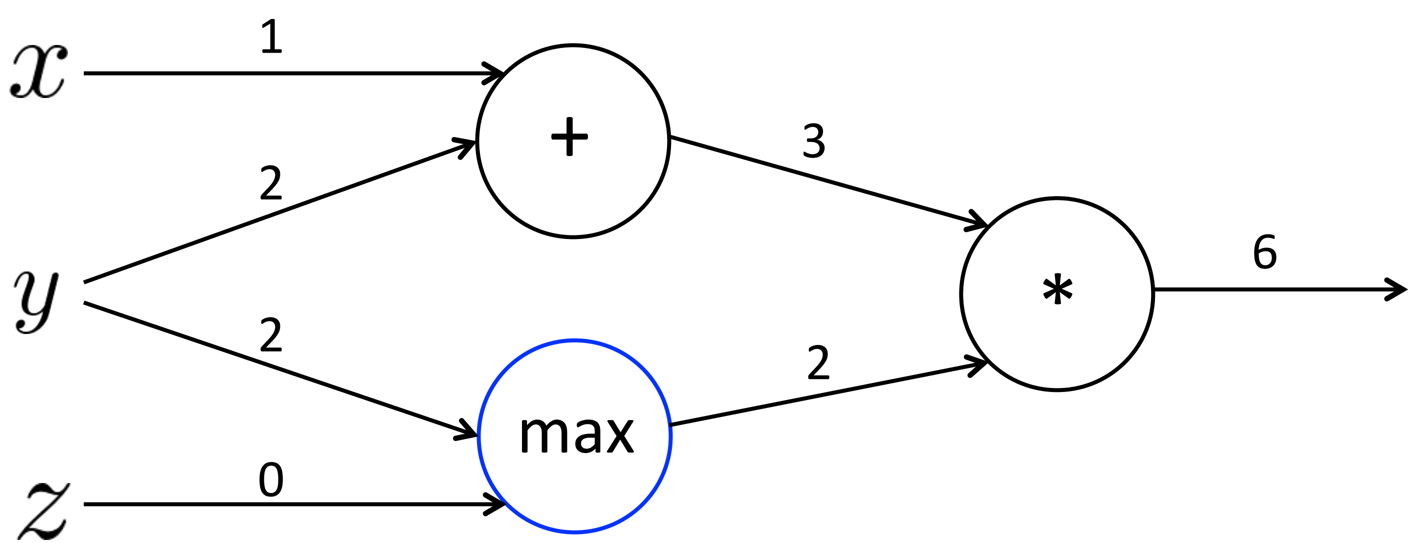
$$f(x,y,z) = (x+y)\max(y,z)$$
$$x = 1, y = 2, z = 0$$

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

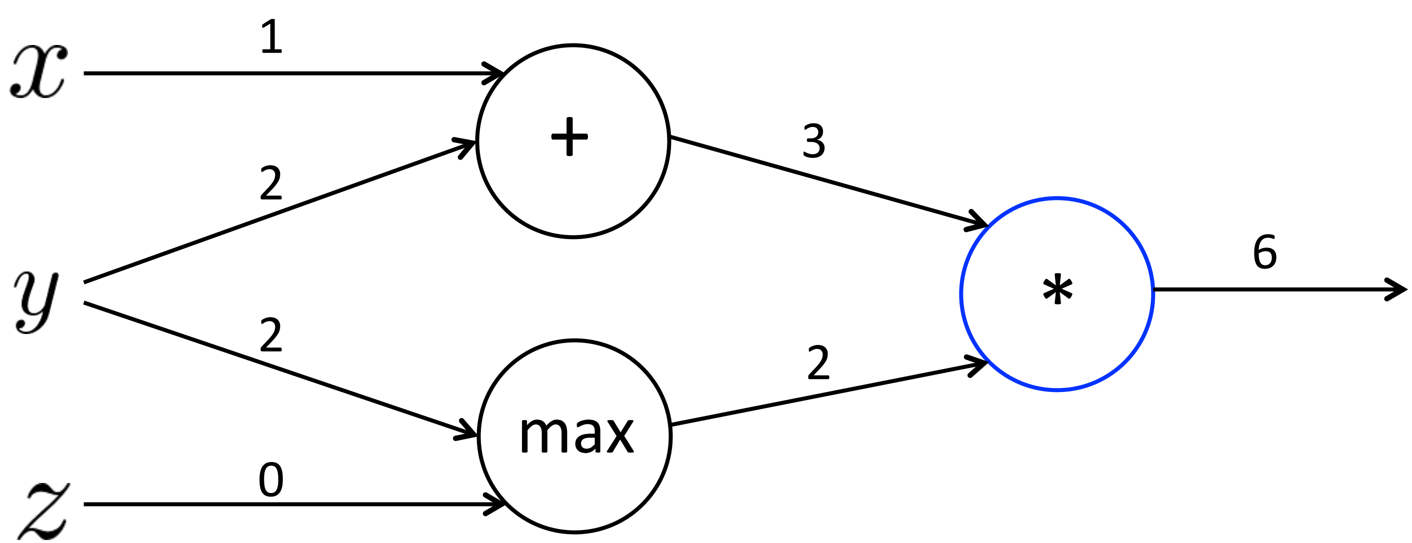
$$f(x, y, z) = (x + y)\max(y, z)$$
$$x = 1, y = 2, z = 0$$

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

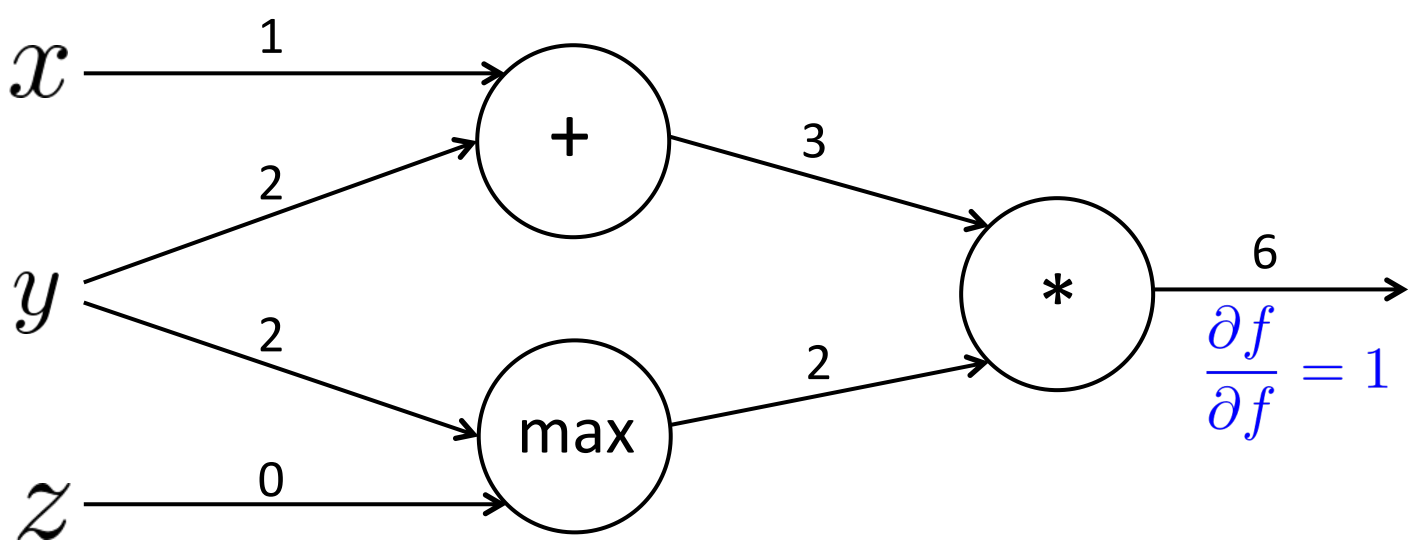
$$f(x, y, z) = (x + y)\max(y, z)$$
$$x = 1, y = 2, z = 0$$

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

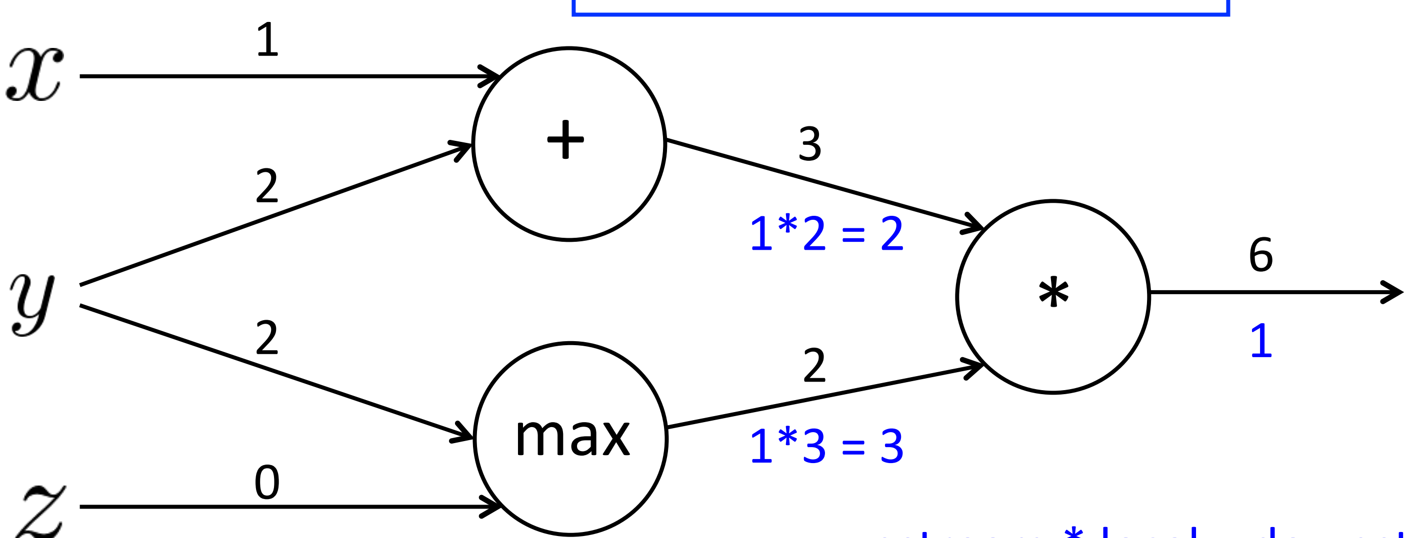
$$f(x, y, z) = (x + y)\max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

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

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$
$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$
$$\boxed{\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3}$$



upstream * local = downstream

An Example

$$\boxed{\begin{aligned} &f(x, y, z) = (x + y)\max(y, z) \\ &x = 1, y = 2, z = 0 \end{aligned}}$$

Forward prop steps

$$a = x + y$$

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

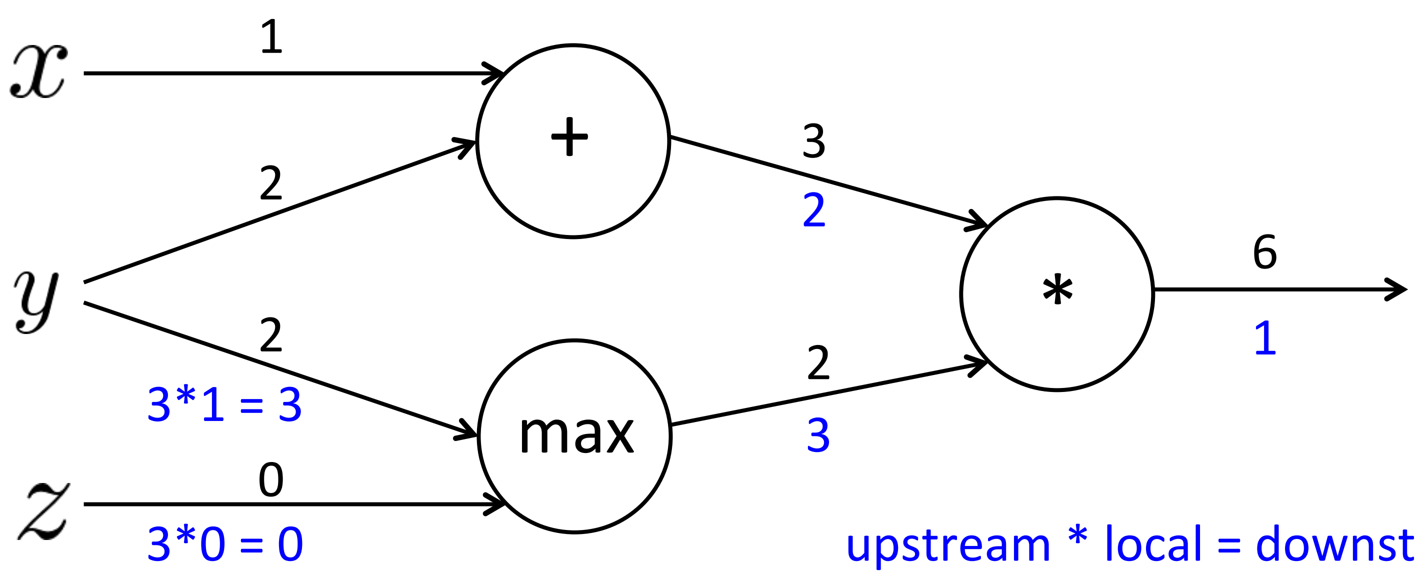
$$f = ab$$

Local gradients

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

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

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



upstream * local = downstream

An Example

$$\boxed{\begin{array}{l} f(x,y,z) = (x+y)\max(y,z) \\ x=1, y=2, z=0 \end{array}}$$

Forward prop steps

$$a = x + y$$

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

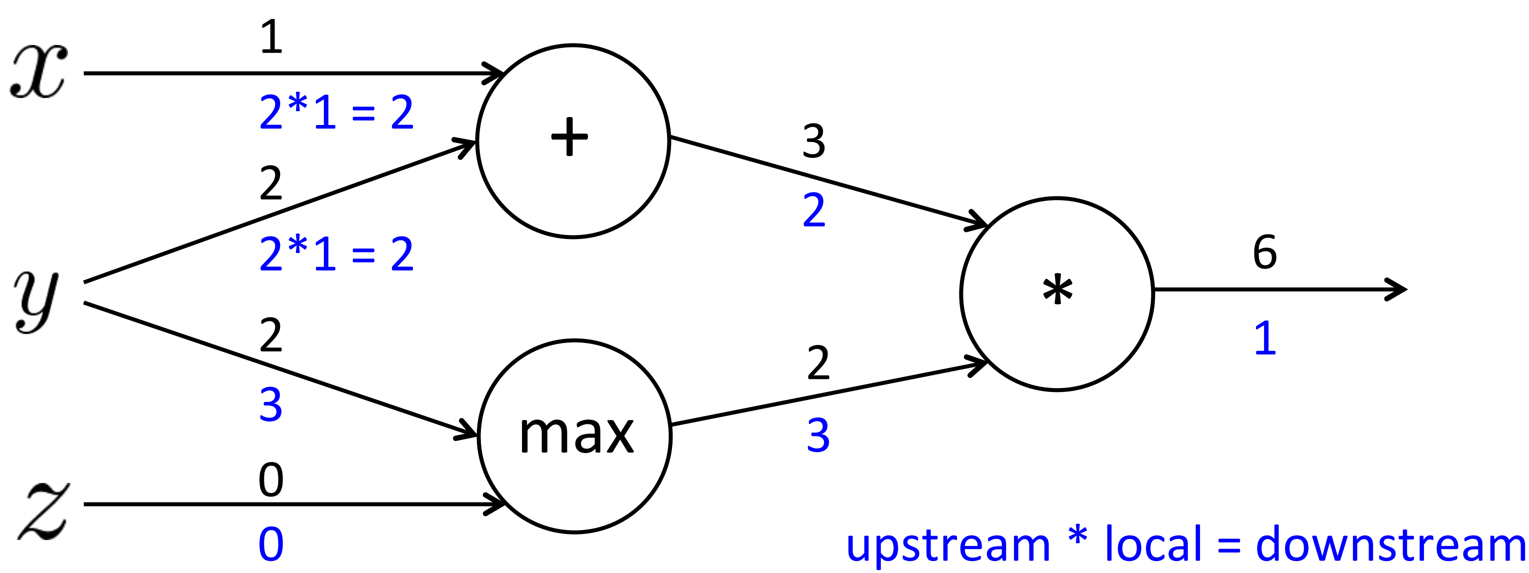
$$f = ab$$

Local gradients

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

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

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



An Example

$$\boxed{\begin{array}{l} f(x,y,z) = (x+y)\max(y,z) \\ x=1, y=2, z=0 \end{array}}$$

Forward prop steps

$$a = x + y$$

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

$$f = ab$$

Local gradients

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

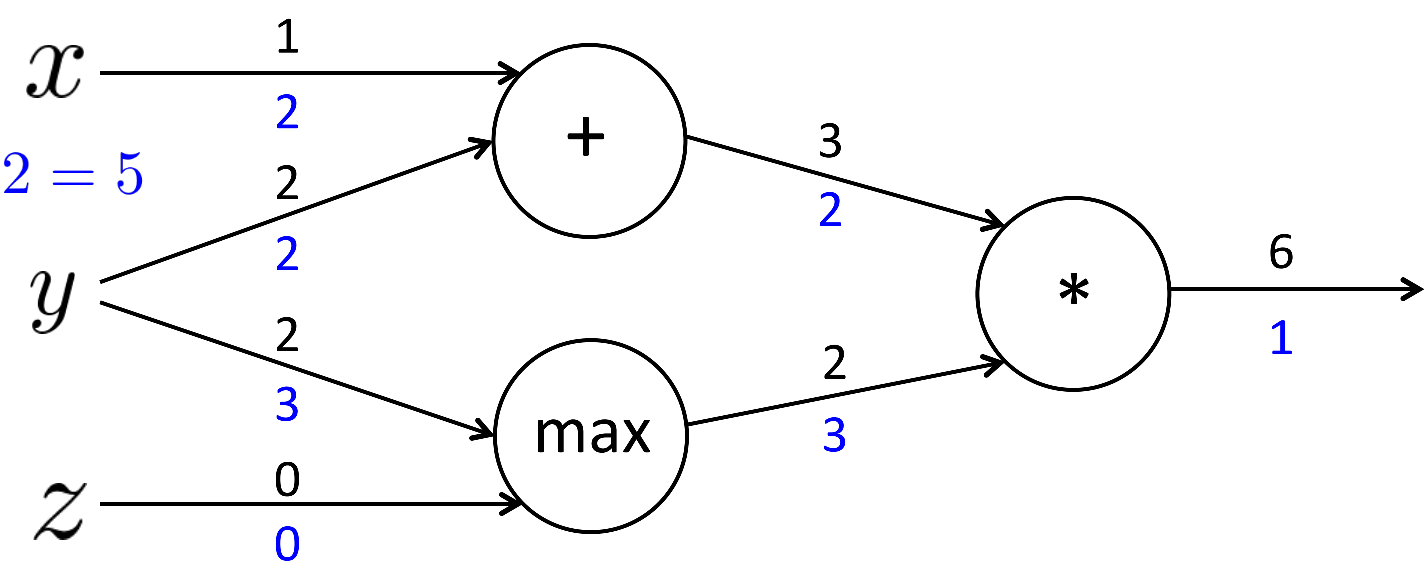
$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

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

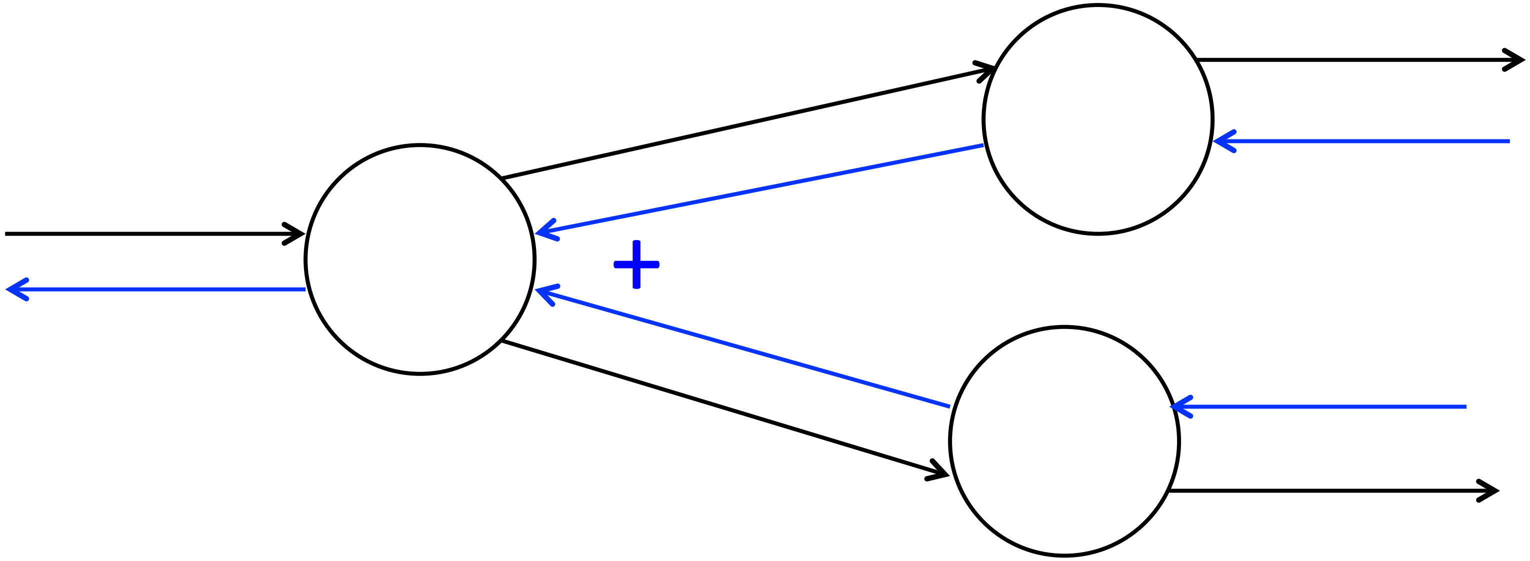
$$\frac{\partial f}{\partial x} = 2$$

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

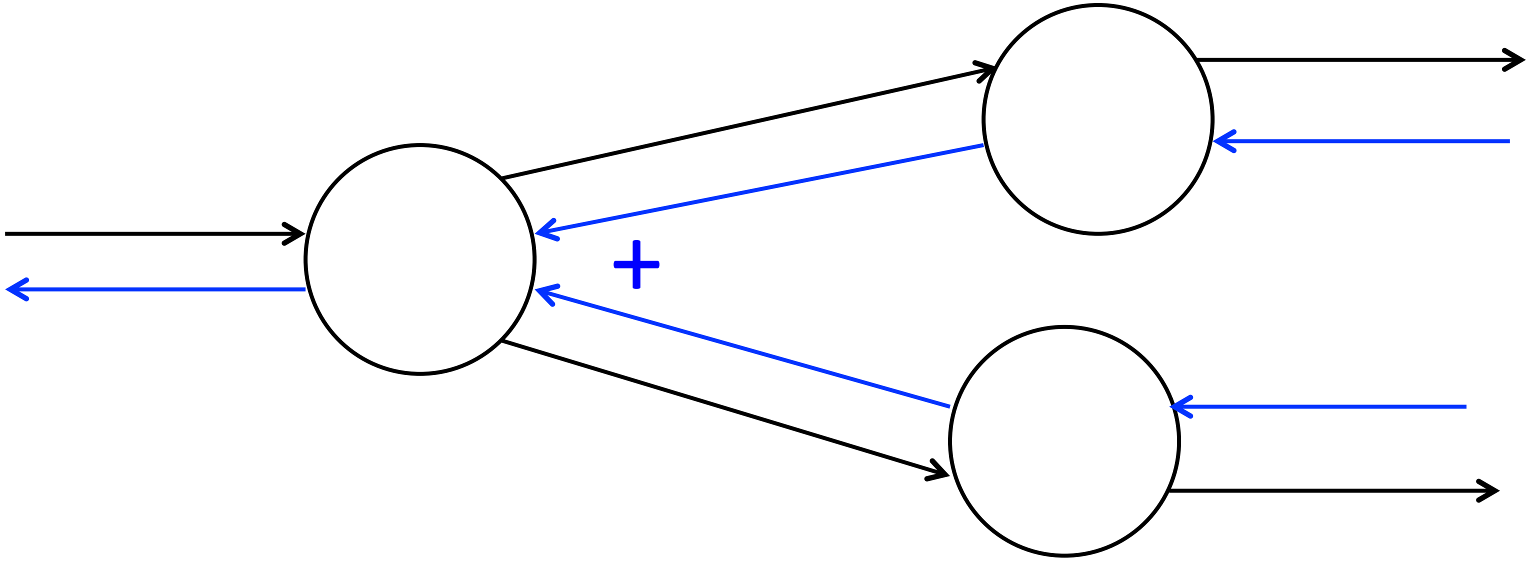
$$\frac{\partial f}{\partial z} = 0$$



Gradients add at branches



Gradients add at branches



$$a = x + y$$

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

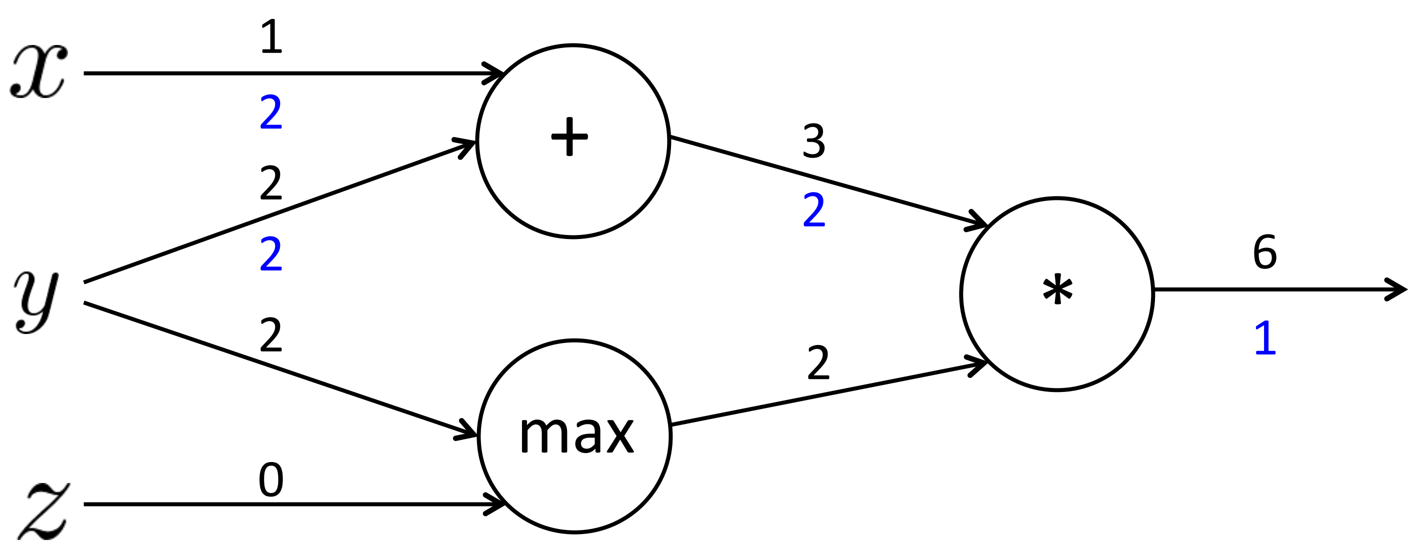
$$f = ab$$

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

Node Intuitions

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

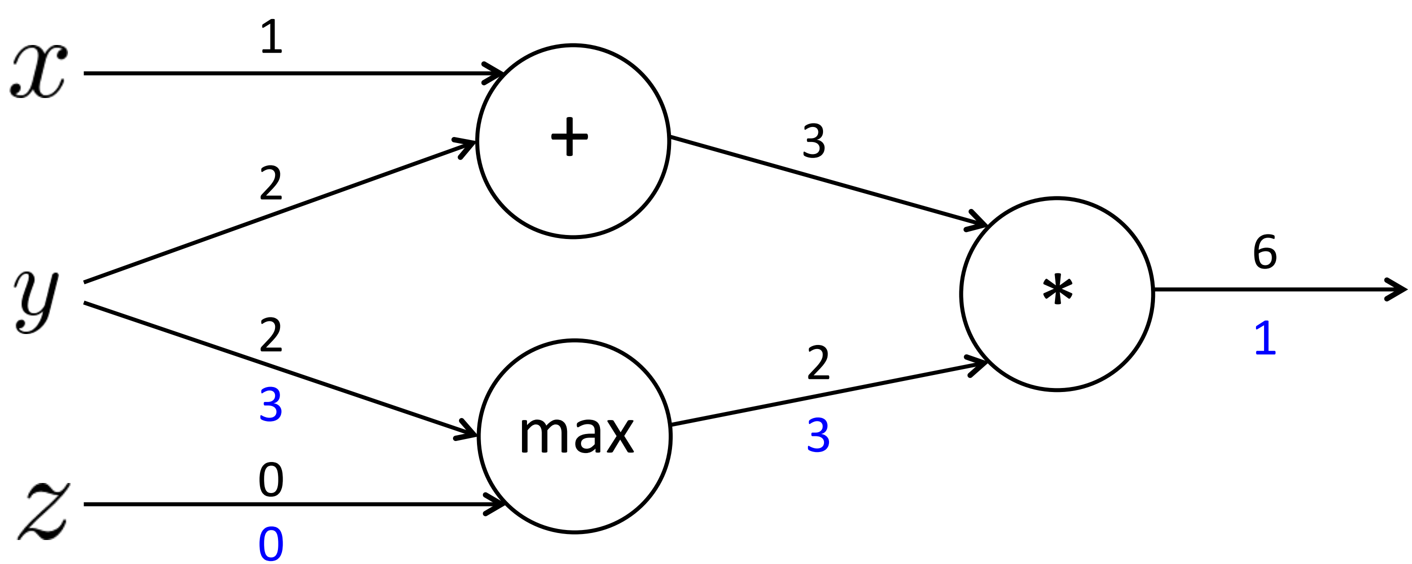
- + "distributes" the upstream gradient



Node Intuitions

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

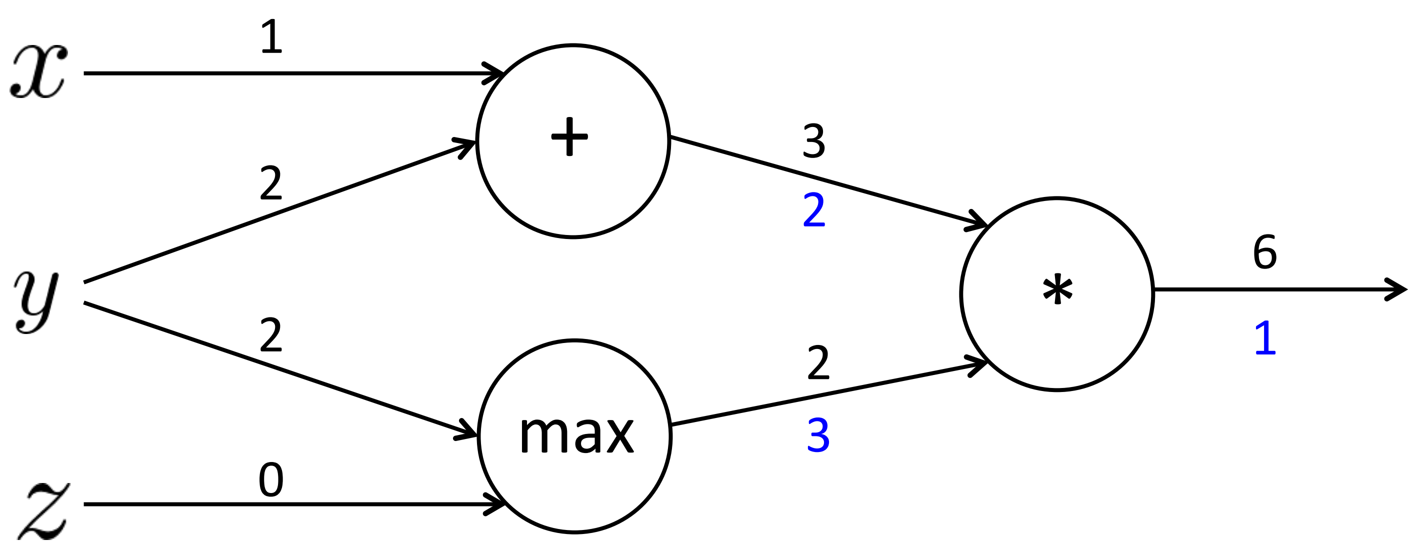
- + "distributes" the upstream gradient
- max "routes" the upstream gradient



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

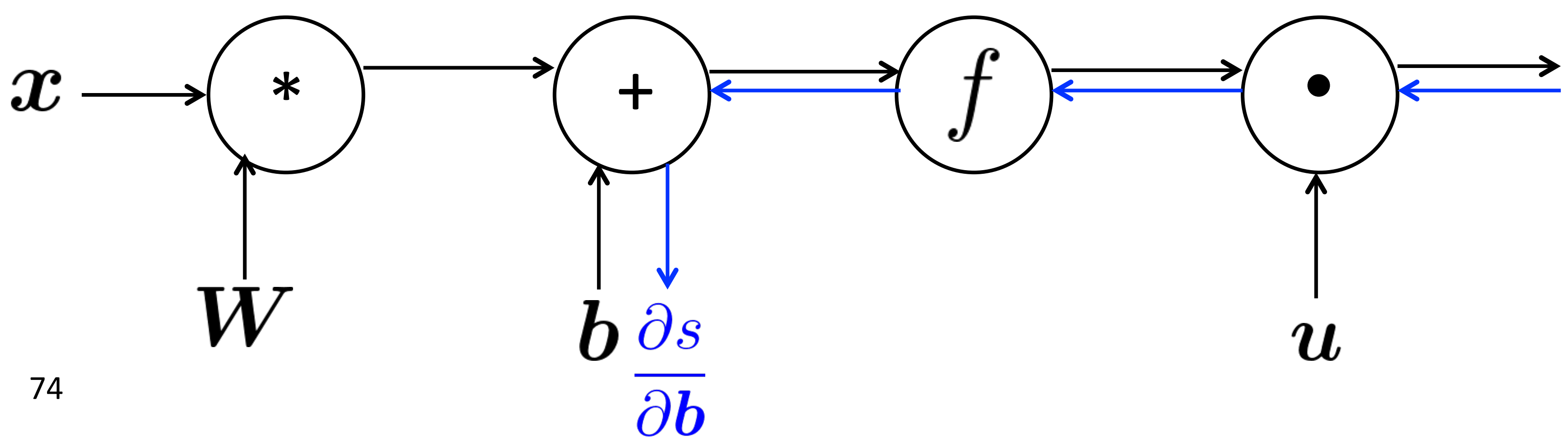
- Incorrect way of doing backprop:
  - First compute $\frac{\partial s}{\partial \boldsymbol{b}}$

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

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

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

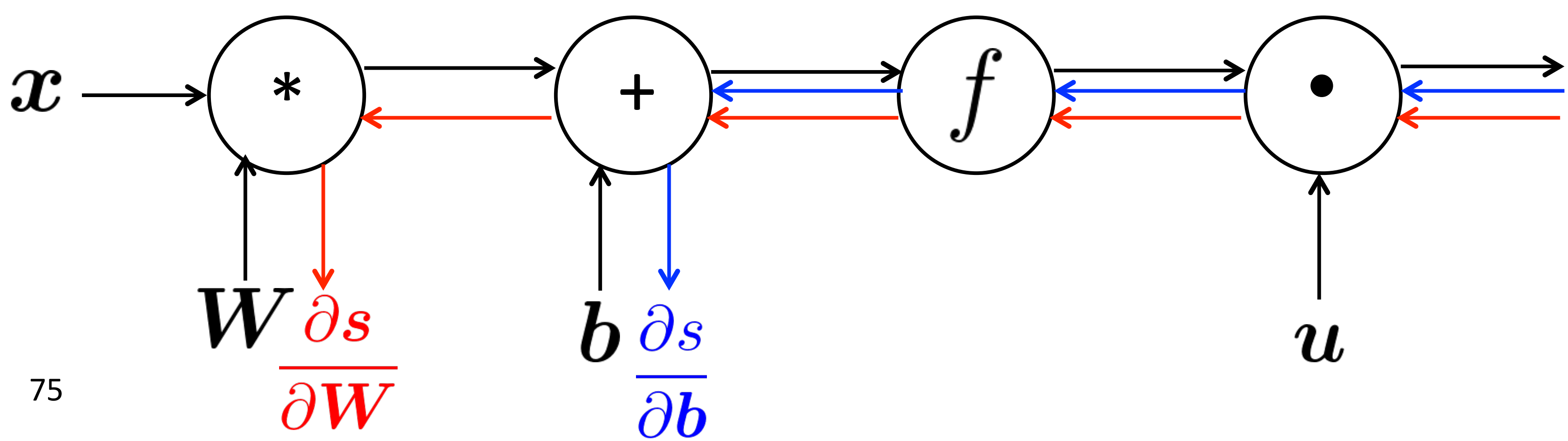
$$\boldsymbol{x} \quad \text{(input)}$$



Efficiency: compute all gradients at once

- Incorrect way of doing backprop:
  - First compute $\frac{\partial s}{\partial \boldsymbol{b}}$
  - Then independently compute $\frac{\partial s}{\partial \boldsymbol{W}}$
  - Duplicated computation!

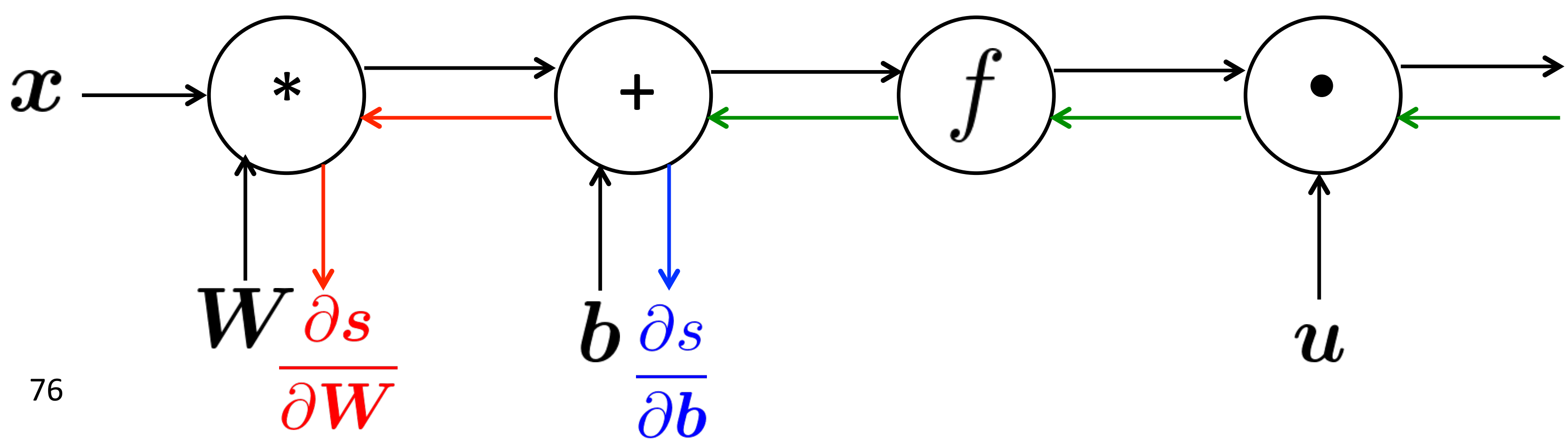
$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$



Efficiency: compute all gradients at once

- Correct way:
  - Compute all the gradients at once
  - Analogous to using $\boldsymbol{\delta}$ when we computed gradients by hand

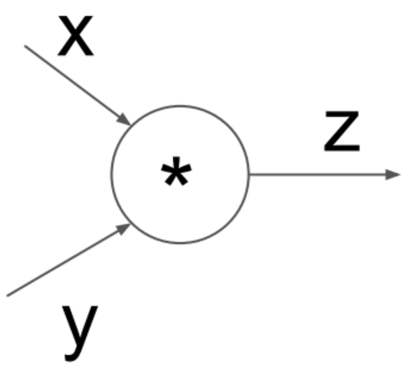
$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$



Backprop Implementations

```python
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



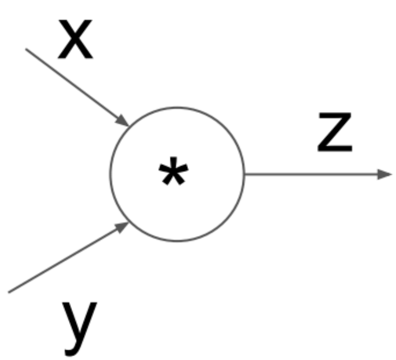
(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        return z
    def backward(dz):
        # dx = ... #todo
        # dy = ... #todo
        return [dx, dy]
```

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

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

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