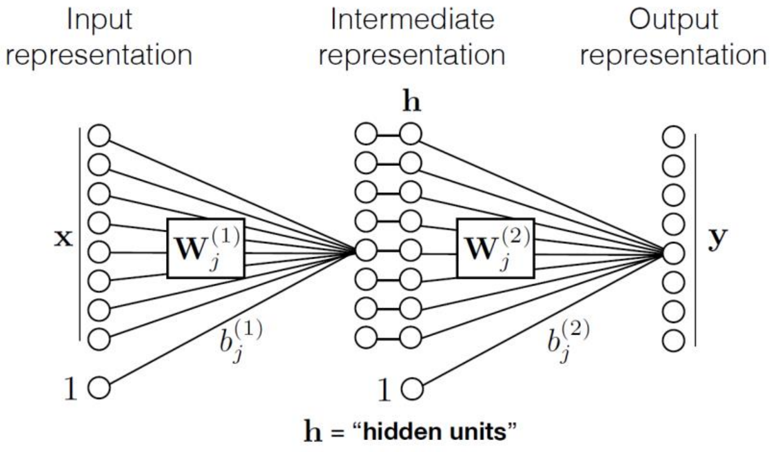
Training Neural Networks

Training Deep Neural Networks

- Minimizing the loss function
  1. Initialize all weights in the network parameters to Gaussian or uniform random values (mean=0, σ). Bias terms are initialized to zeros
  2. Forward pass: Apply network functions to input data.
  3. Compute the loss
  4. Backward pass: Take the derivative of the loss with respect to the weights and move in the direction negative to the gradient (Backpropagation algorithm)
  5. Update the weights
  6. Go back to step 2 and repeat until a desired number of iterations (*epochs*) or termination criteria is reached.



$$\boxed{\theta^* = \arg\min_{\theta} \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}}$$

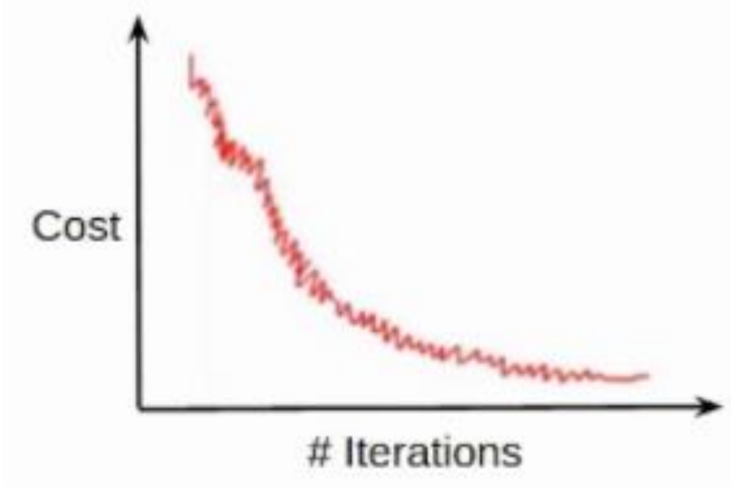
One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \left. \frac{\partial J(\theta)}{\partial \theta} \right|_{\theta=\theta^t}$$

learning rate

Updating the Weights, When?

- Our goal is to minimize overall loss function **J**, which is sum of individual losses over each example
- Updating weights obviously must occur after computing the loss function (i.e., errors) as this allows us to compute $\frac{\partial J}{\partial \theta}$
- We can update the weights after each training sample…Takes a long time and noisy



$$\theta^* = \arg\min_\theta \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

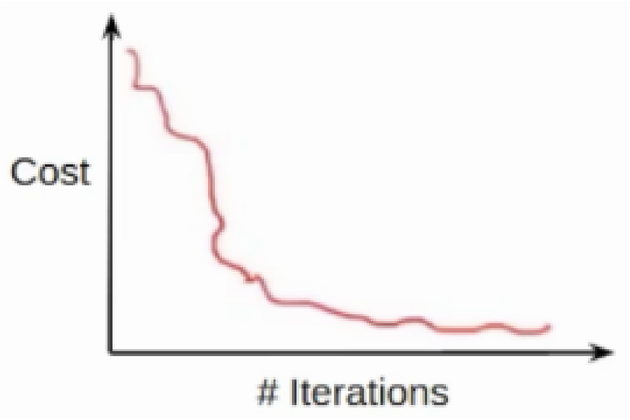
One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \frac{\partial J(\theta)}{\partial \theta}\bigg|_{\theta=\theta^t}$$

learning rate

Mini-batch gradient descent

- An alternate approach is to compute overall error for a number of samples in the dataset and then update the weights
- This approach is often used in practice and is called ‘mini batch gradient descent’. Using this strategy, the training dataset is divided into several smaller batches. Gradient updates once for each batch



Mini-batch gradient descent

- Recommended size for minibatch is the largest that can fit available memory

Goyal, Priya, et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour." arXiv preprint arXiv:1706.02677 (2017)

- The extreme case is where the whole dataset is considered one batch “ batch gradient dissent” offers the smoothest decay but can reduce generalization performance

Updating the Weights, How?

- Computing $\frac{\partial J}{\partial \theta}$ is accomplished using the chain rule in a process that moves backward from the output of the network to its input
- This algorithm is known as

  "<u>backpropagation</u>"

$$\boxed{\theta^* = \arg\min_\theta \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}}$$

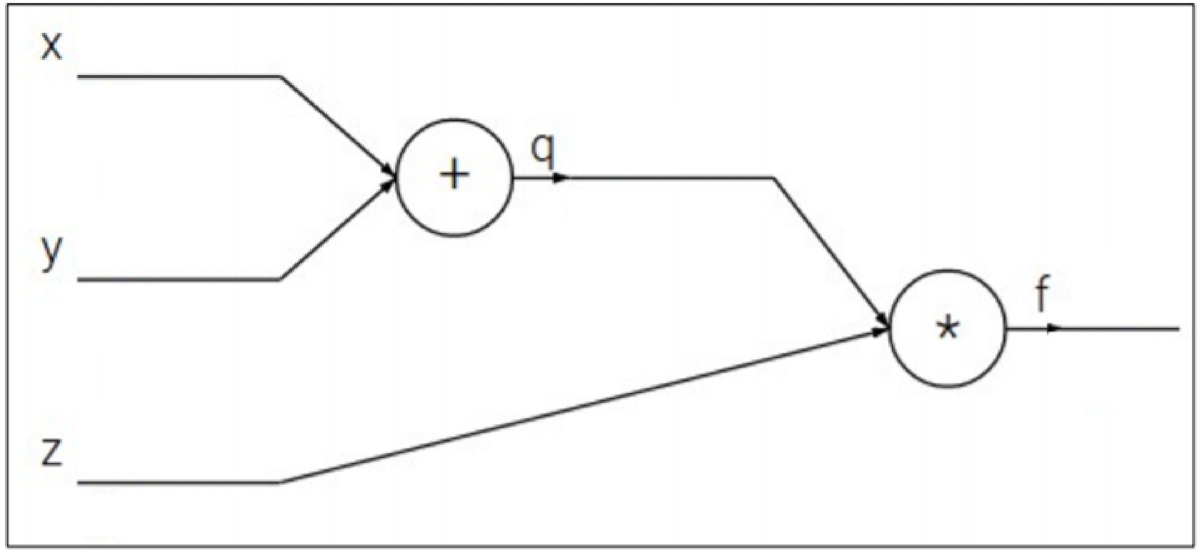
One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \left.\frac{\partial J(\theta)}{\partial \theta}\right|_{\theta=\theta^t}$$

learning rate

Backpropagation: Simple Example

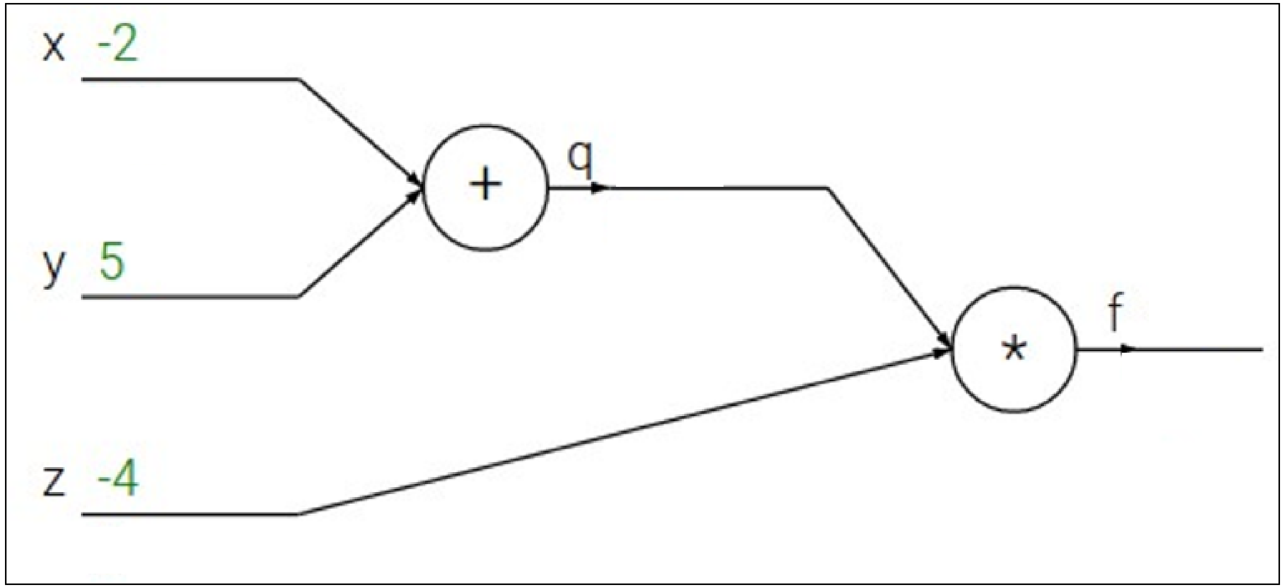
$$f(x, y, z) = (x + y) \cdot z$$



Backpropagation: Simple Example

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

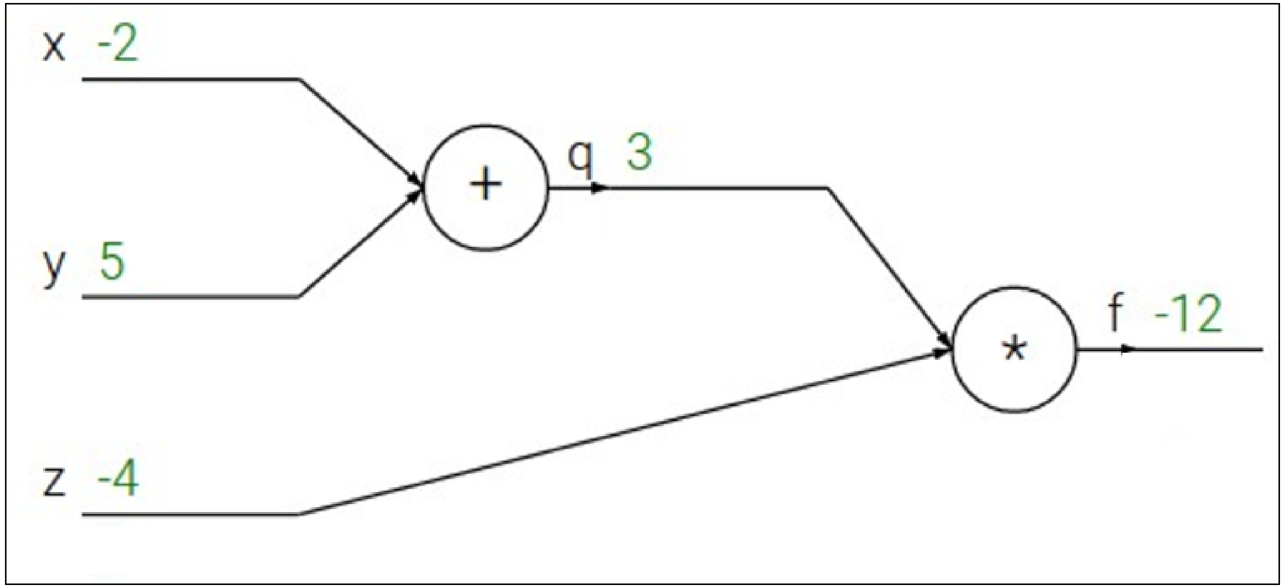
e.g. x = -2, y = 5, z = -4



Backpropagation: Simple Example

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

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



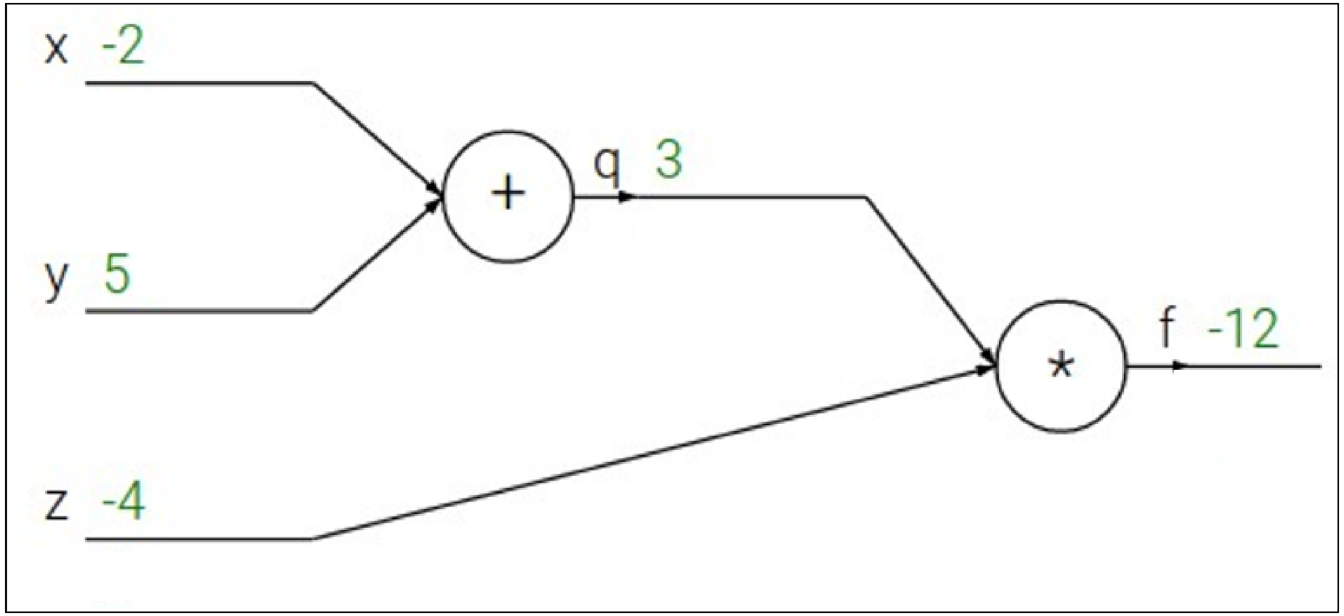
1. Forward pass: Compute outputs

$$q = x + y \quad f = q \cdot z$$

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = q \cdot z$$

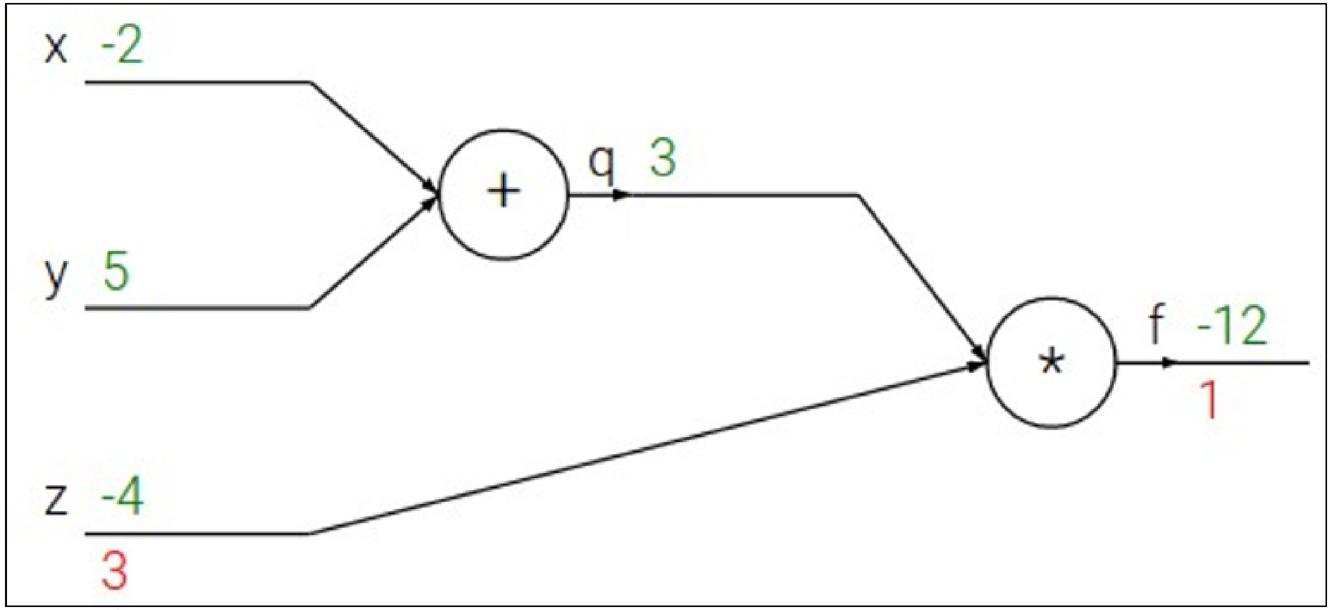
2. Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

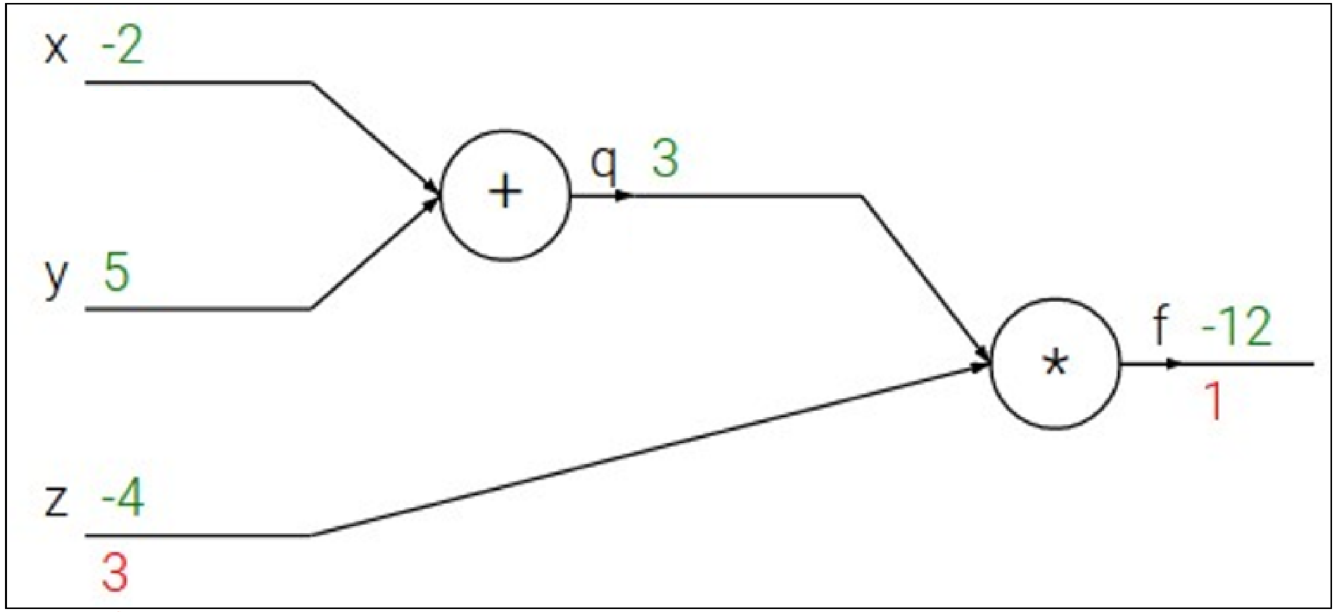
$$\boxed{\frac{\partial f}{\partial z}}$$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

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

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

Backpropagation: Simple Example

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

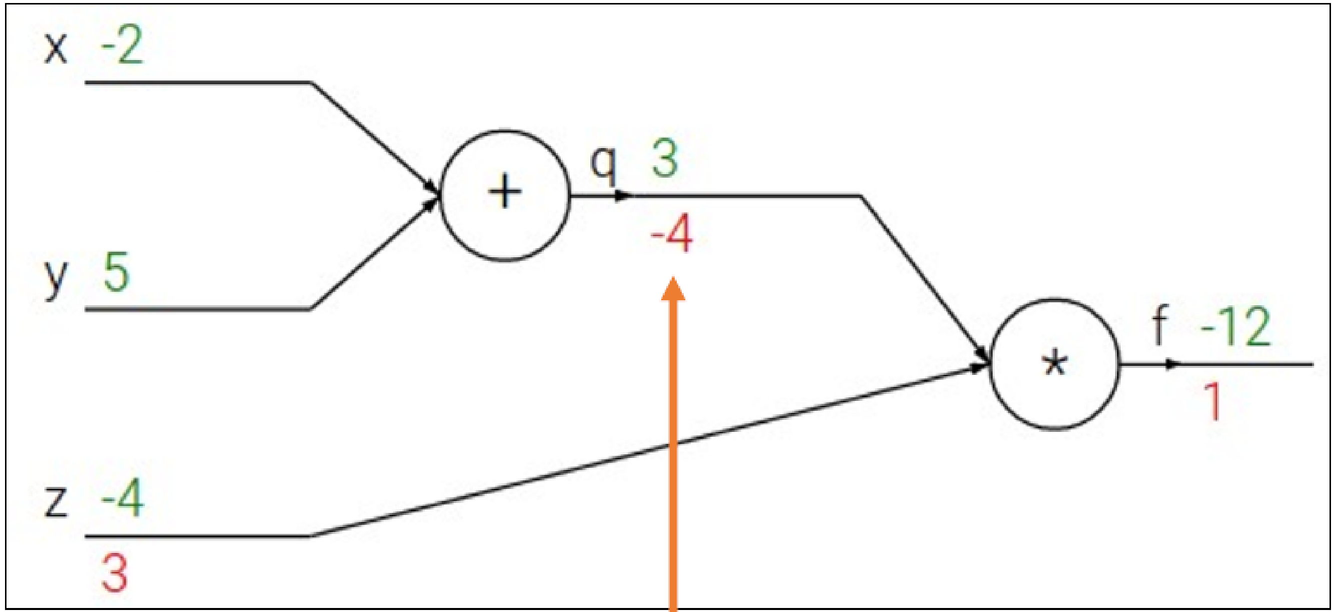
e.g. x = -2, y = 5, z = -4

1. Forward pass: Compute outputs

$$q = x + y \qquad f = q \cdot z$$

2. Backward pass: Compute derivatives

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

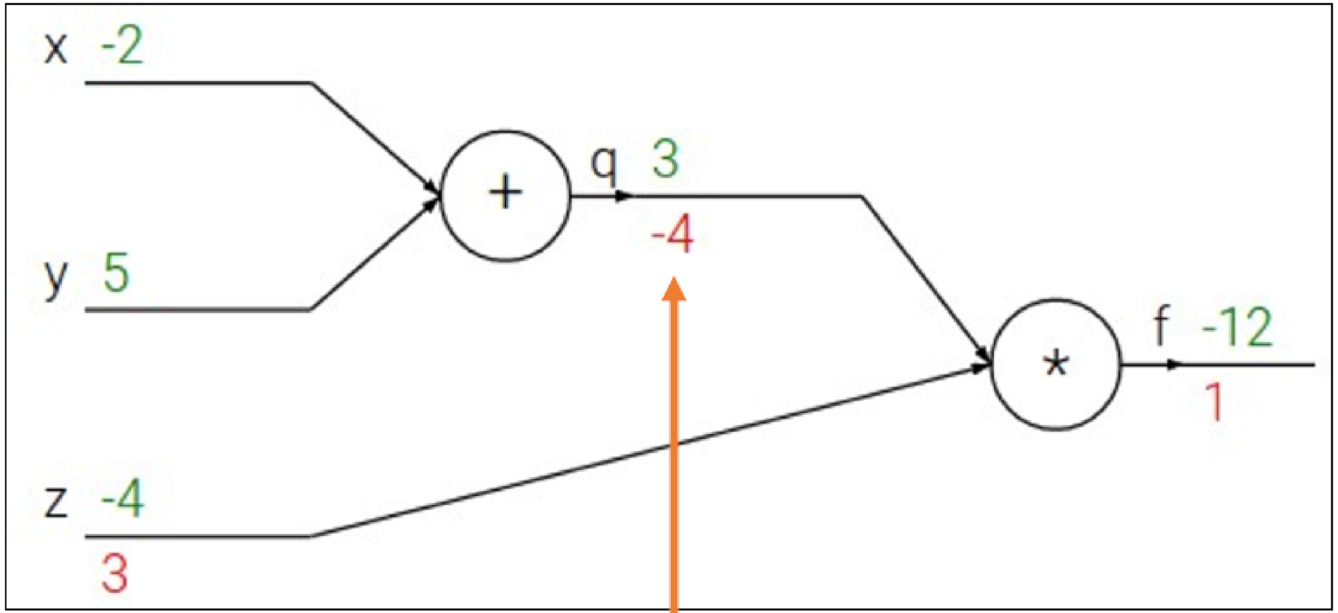


$$\frac{\partial f}{\partial q}$$

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

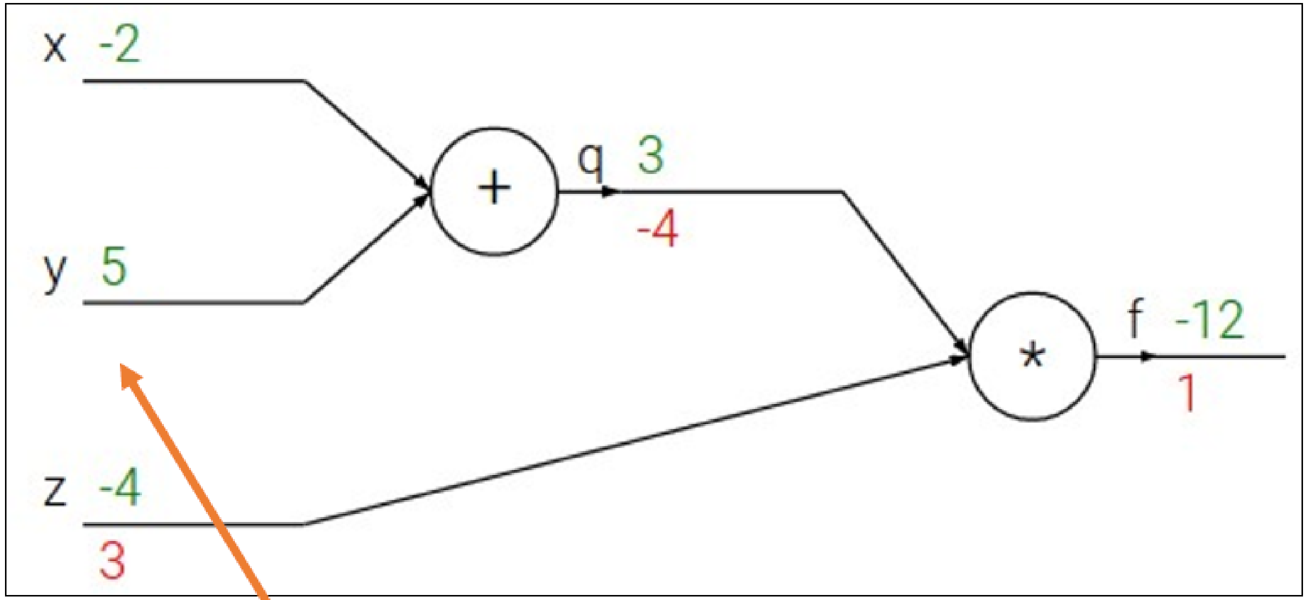
$$\boxed{\frac{\partial f}{\partial q} = z}$$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = q \cdot z$$

2. Backward pass: Compute derivatives

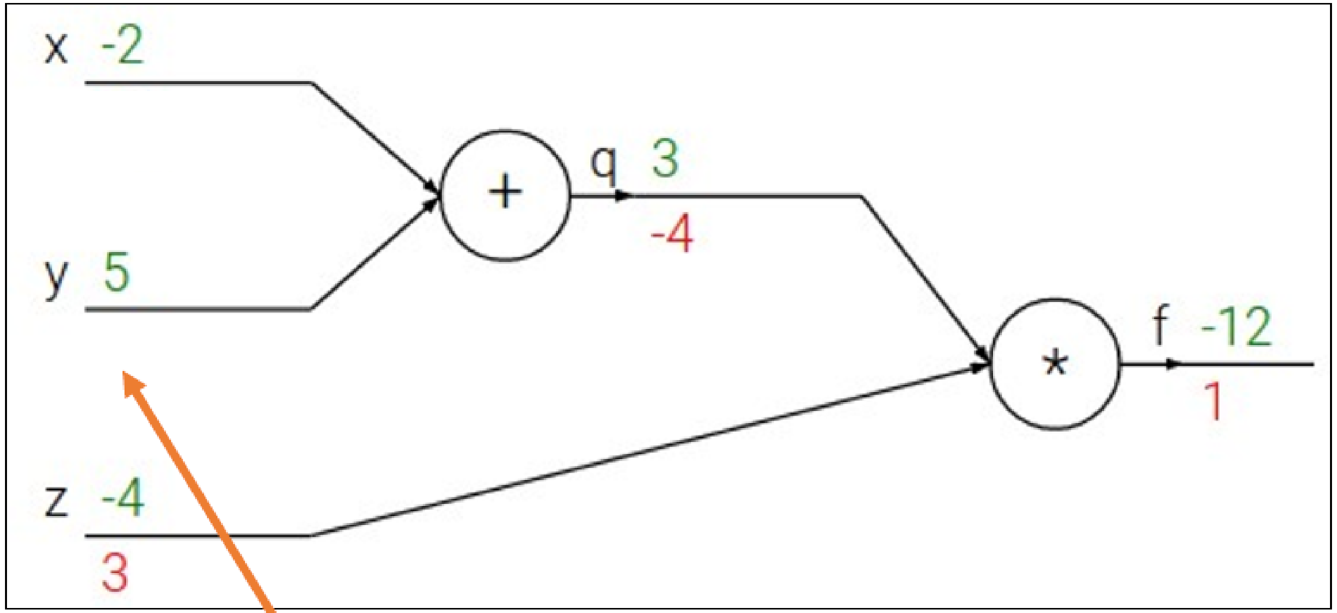
Want: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

$$q = x + y \qquad f = q \cdot z$$

Chain Rule

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

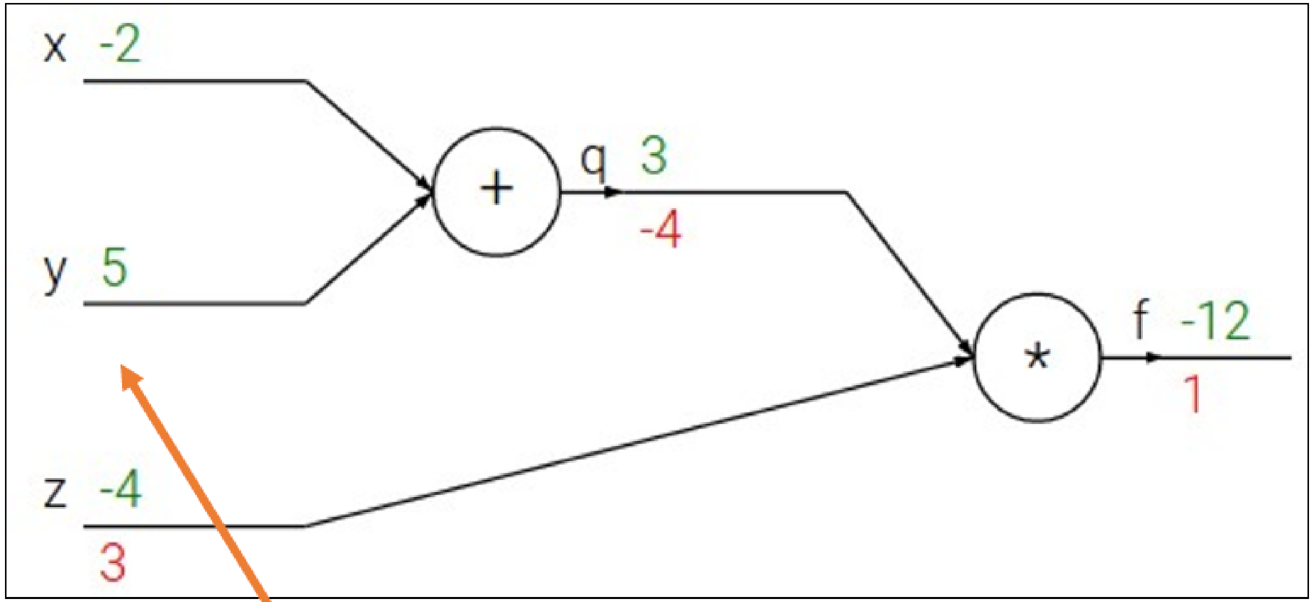
2. Backward pass: Compute derivatives

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

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

Chain Rule

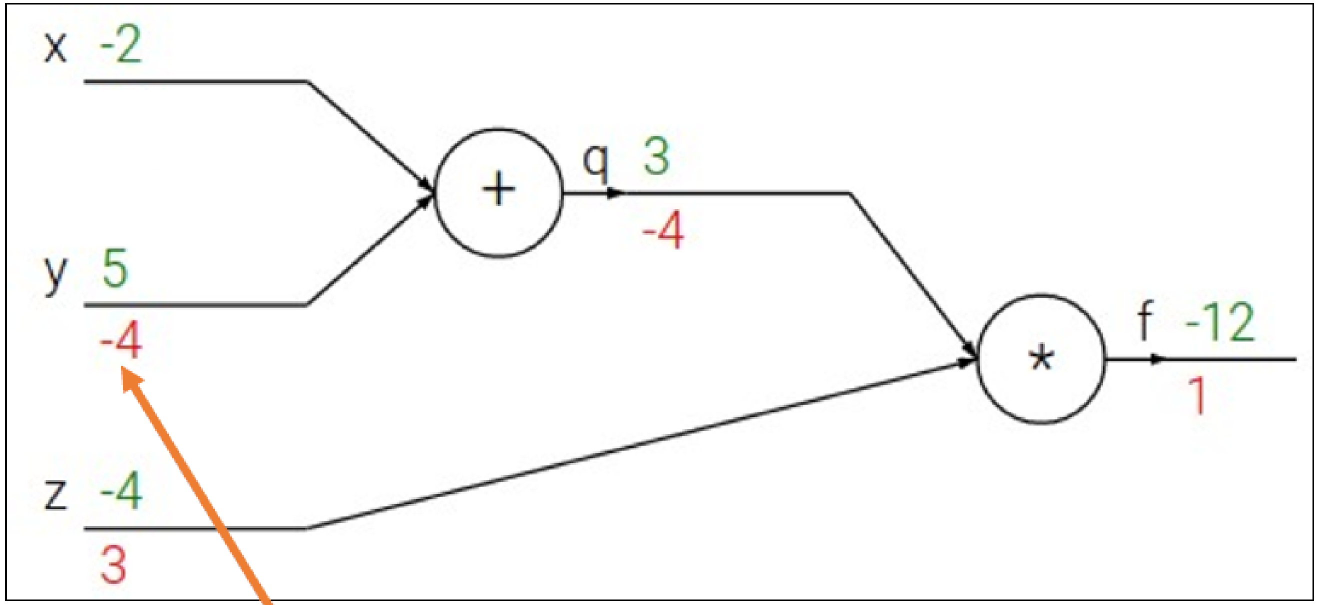
$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial q}{\partial y}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial y} = 1}$$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

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

Chain Rule

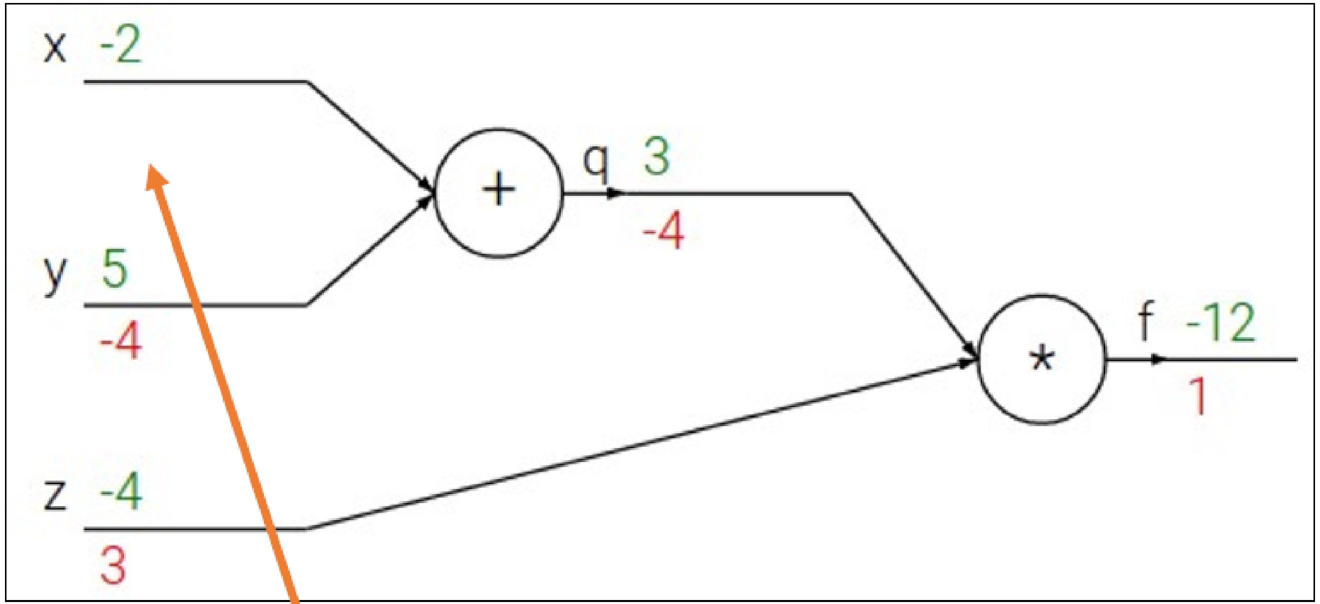
$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial q}{\partial y}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial y} = 1}$$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

2. Backward pass: Compute derivatives

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

Chain Rule

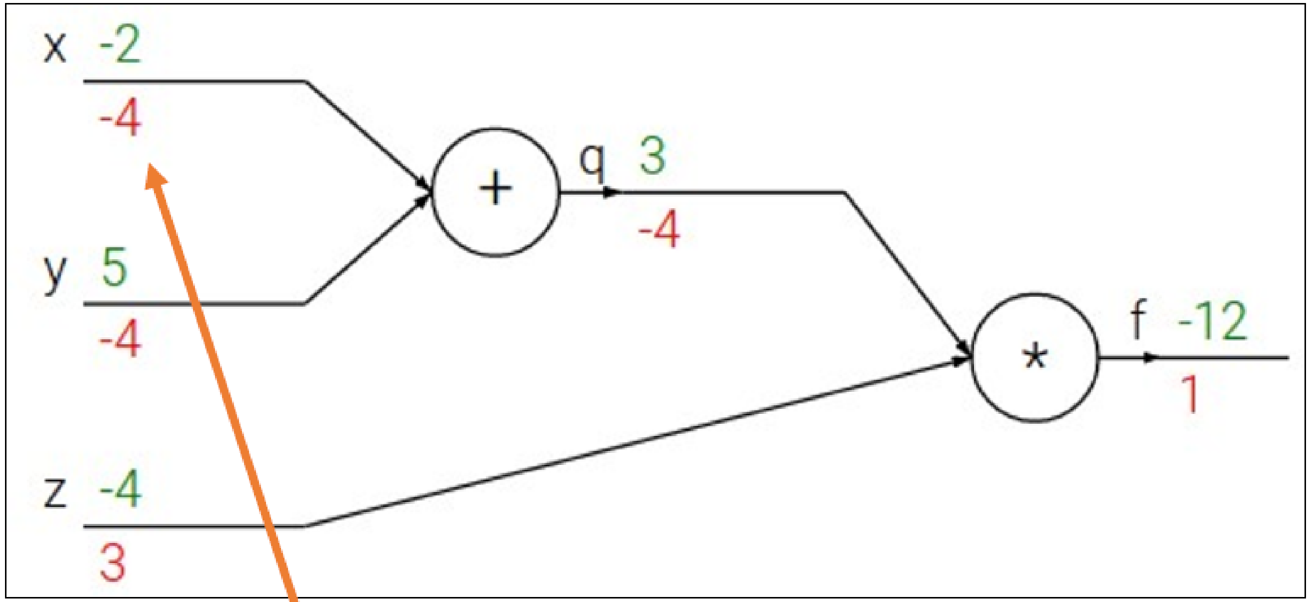
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial x} = 1}$$

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

Backpropagation: Simple Example

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

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



1. Forward pass: Compute outputs

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

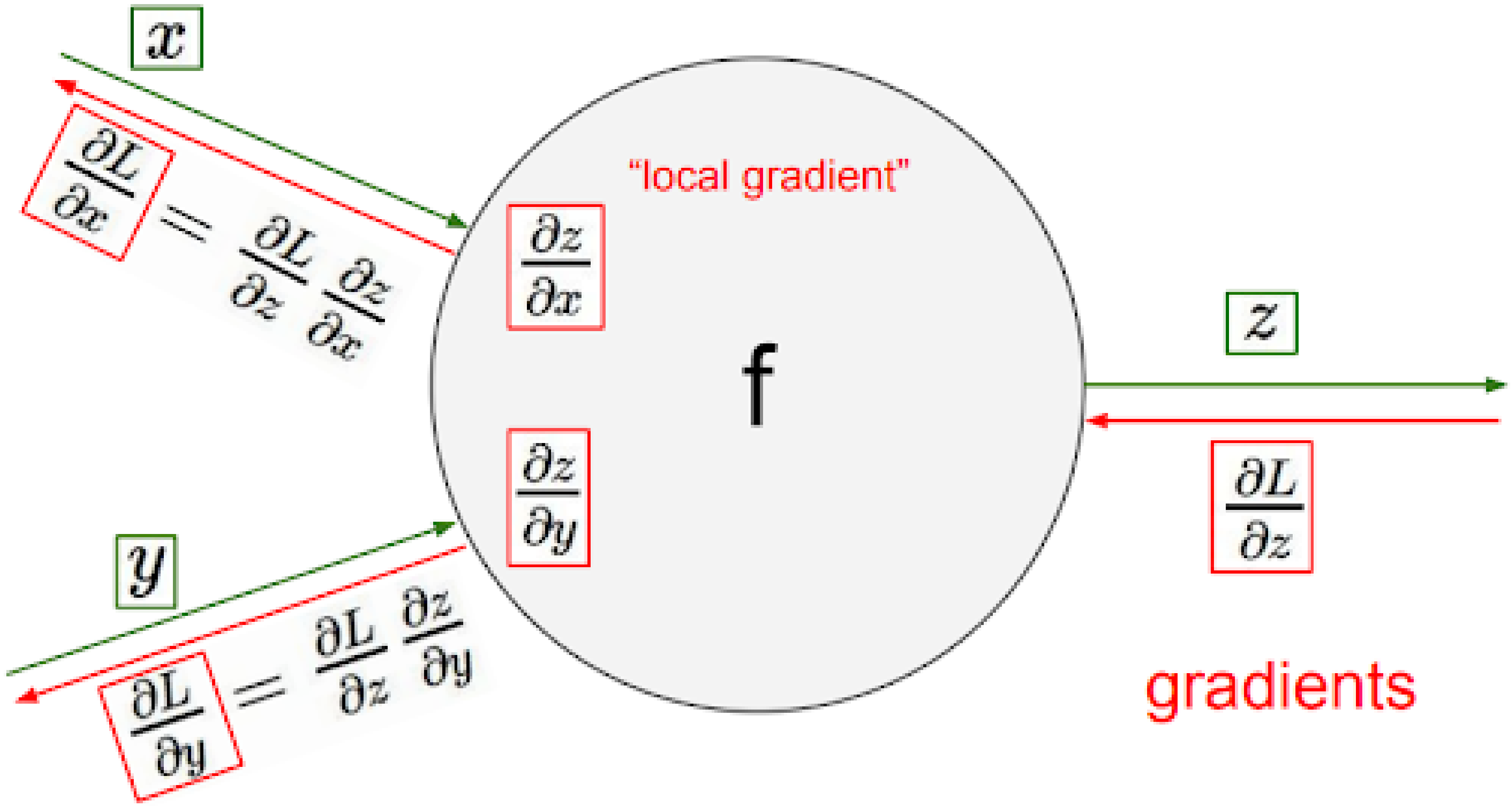
2. Backward pass: Compute derivatives

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

Chain Rule

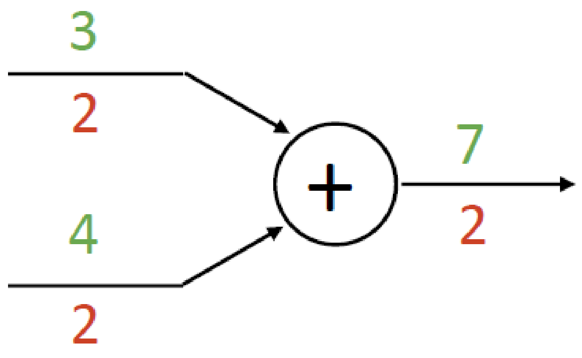
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}} \qquad \boxed{\frac{\partial q}{\partial x} = 1}$$

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

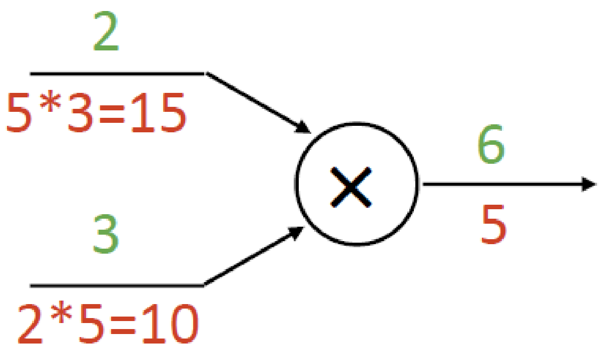

x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Backpropagation

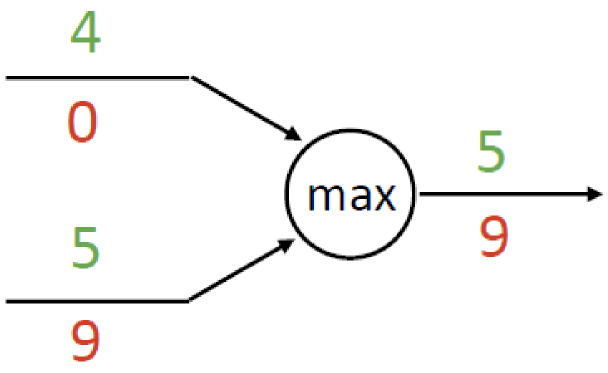
add gate: gradient distributor



mul gate: "swap multiplier"



max gate: gradient router



DNN packages

- These packages automate the process of computing gradients in neural networks.
- Provide implementations for many essential operations in neural networks (*e.g.,* loss functions, activations, layers, regularization, initialization, … etc.)
- The two most popular ones are Pytorch and Tensorflow/Keras

Python and Pytorch Tutorials

- Python tutorial

https://colab.research.google.com/github/cs231n/cs231n.github.io/blob/master/python-colab.ipynb#scrollTo=pua52BGeL9jW

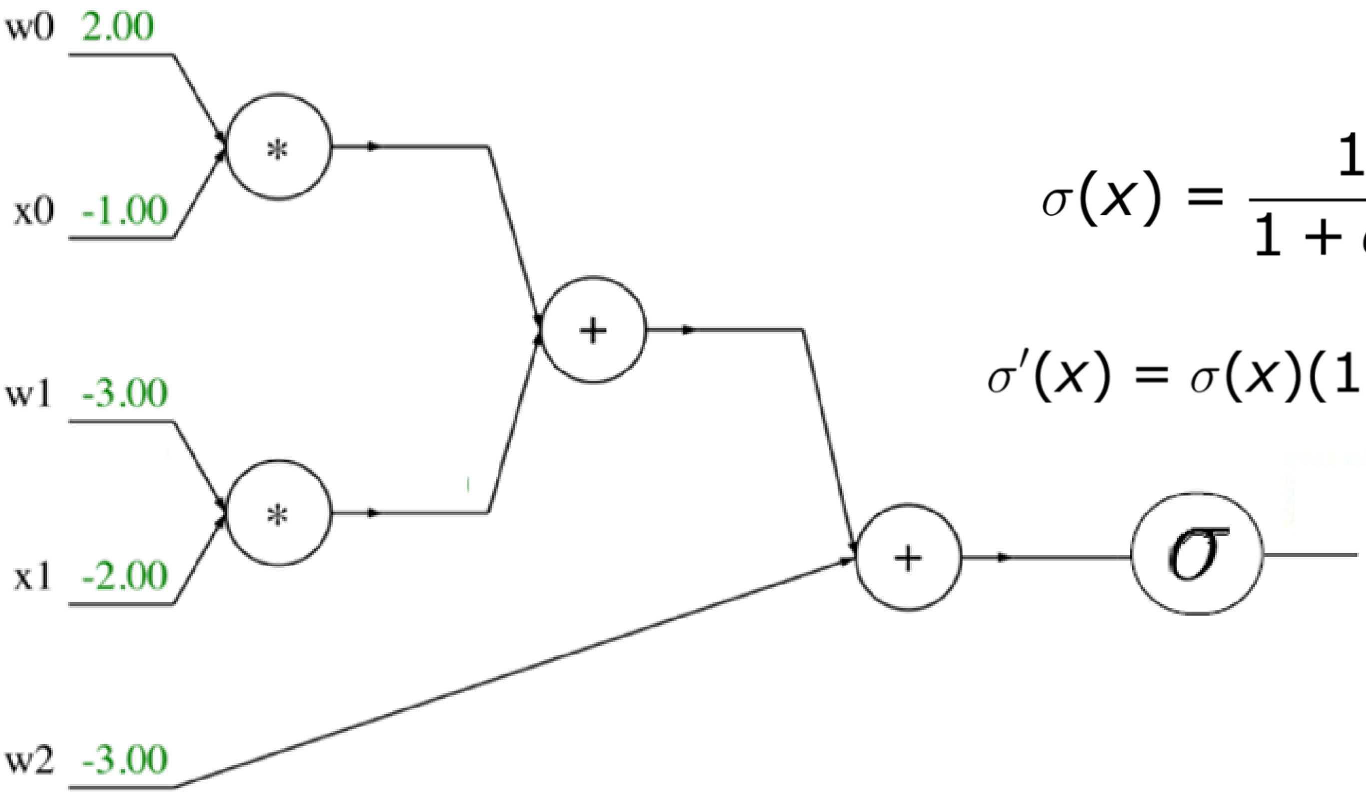
- Pytorch tutorials
- Train an image classifier:

You can run the code below on your browser using google Colab (free GPU for 12 hours) or download it your machine

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#sphx-glr-beginner-blitz-cifar10-tutorial-py

Exercise

- Apply one round of the backpropagation algorithm to the neural network represented by the below computational graph. Computer the updated weight parameters, assume η=1

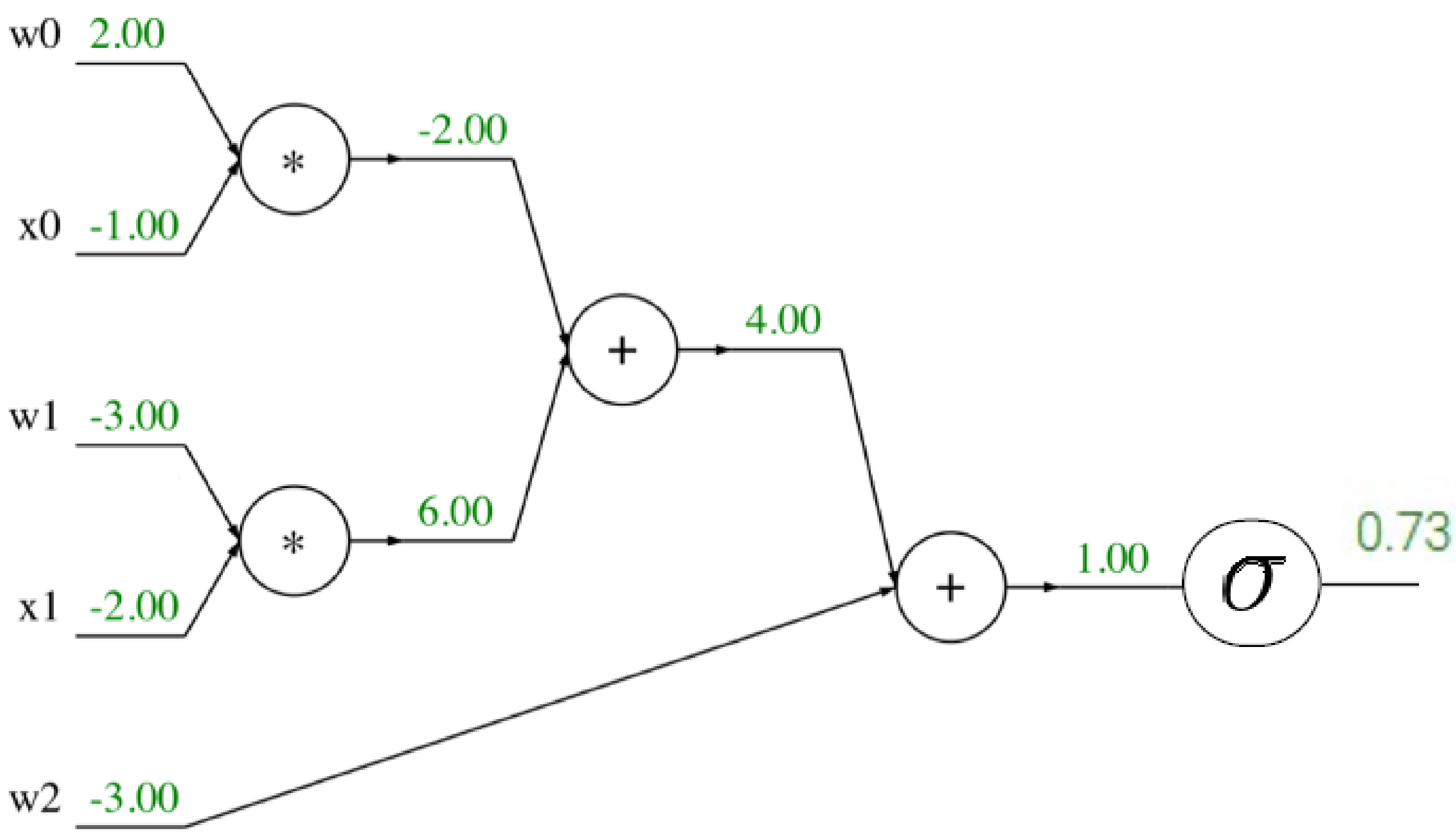


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

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

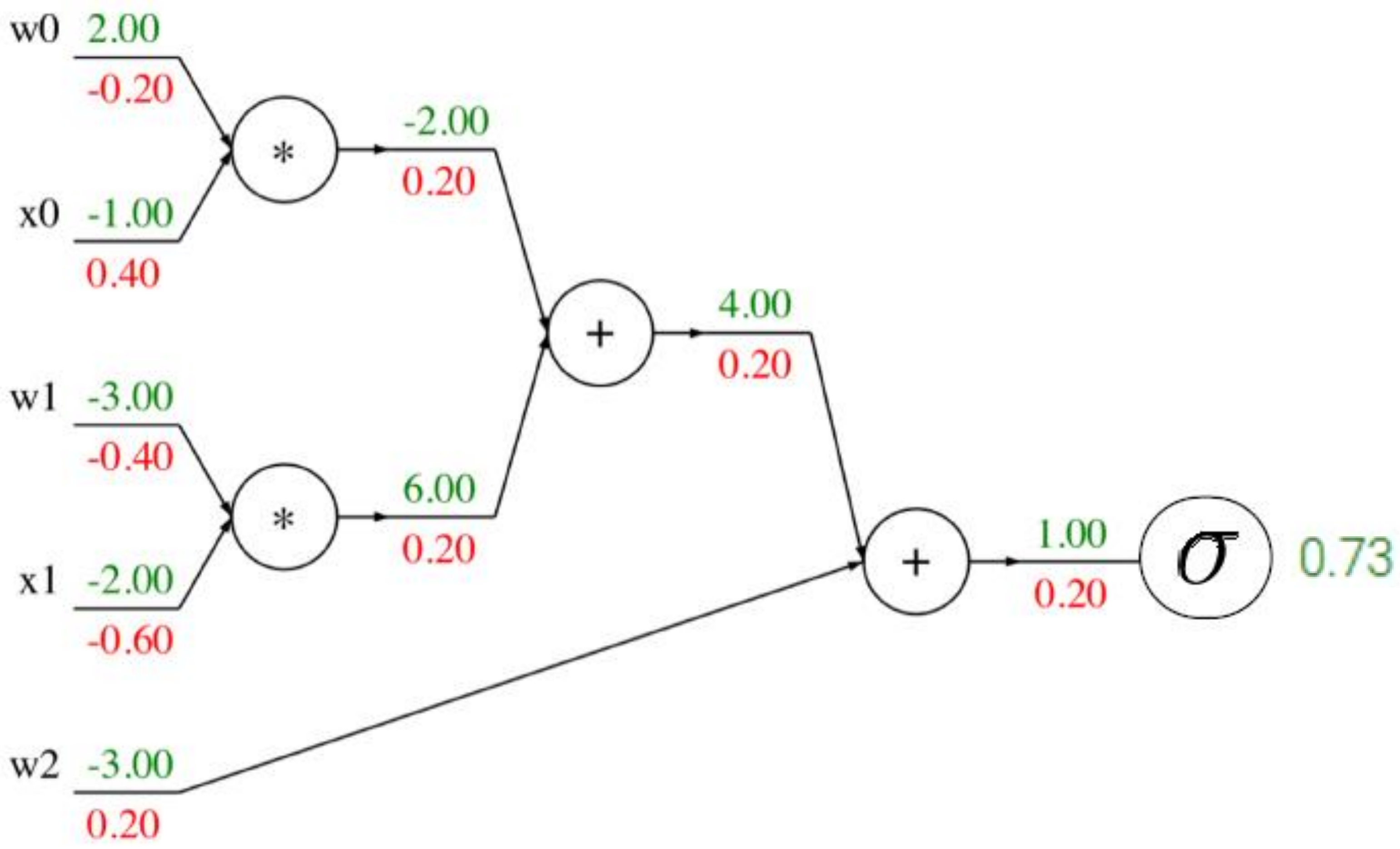
Exercise

Forward pass



Exercise

Backward pass



Weights Updates During Iterations

- As training progresses, each sample shifts the learned parameters towards what achieves its correct label
- As training progresses, the network finds the set of parameters that works well for most or all samples