

# Lecture 4:

# Backpropagation and Neural Networks part 1

# Administrative

A1 is due Jan 20 (Wednesday). ~150 hours left

Warning: Jan 18 (Monday) is Holiday (no class/office hours)

Also note:

Lectures are non-exhaustive.

Read course notes for completeness.

I'll hold make up office hours on Wed Jan20, 5pm @ Gates 259

Where we are...

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

scores function

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

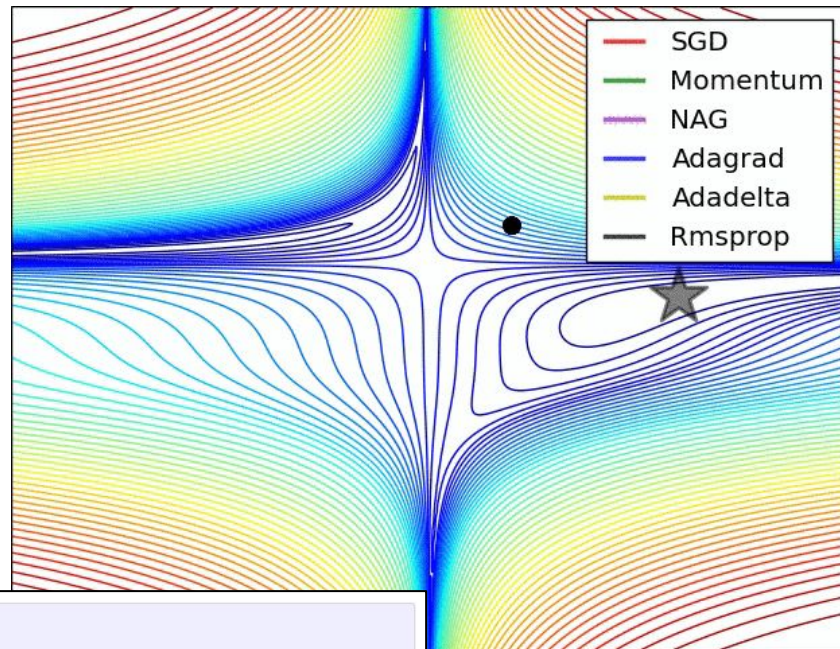
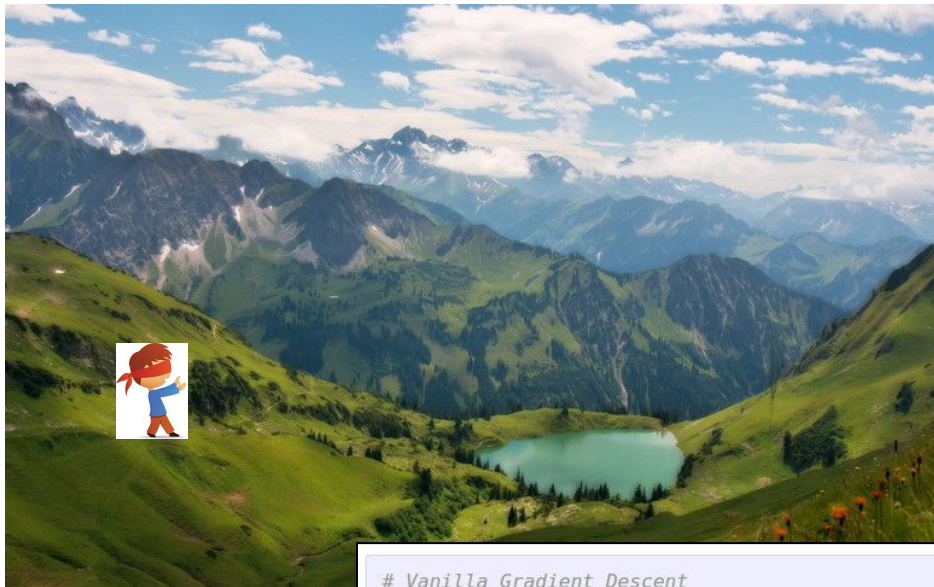
SVM loss

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

data loss + regularization

want  $\nabla_W L$

# Optimization



```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

(image credits  
to Alec Radford)

# Gradient Descent

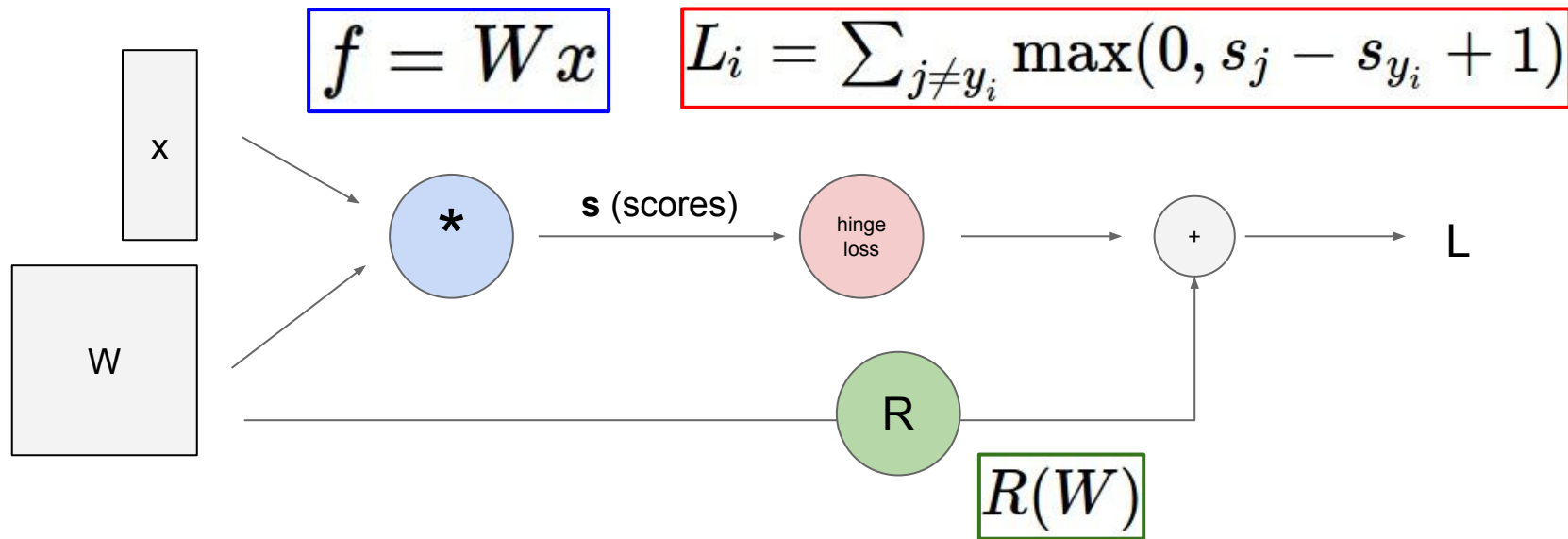
$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

**Numerical gradient:** slow :(, approximate :(, easy to write :)

**Analytic gradient:** fast :), exact :), error-prone :(

In practice: Derive analytic gradient, check your implementation with numerical gradient

# Computational Graph

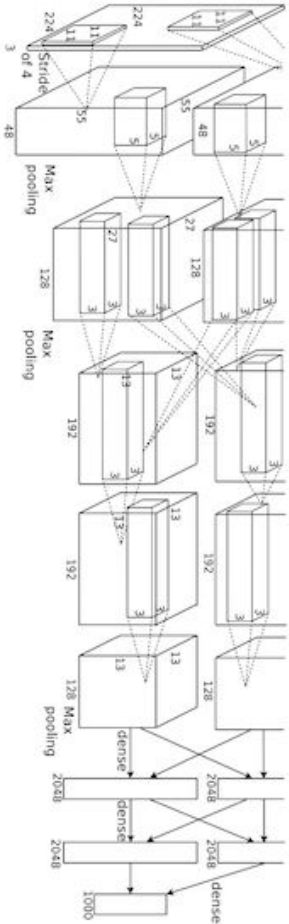


# Convolutional Network (AlexNet)

input image

weights

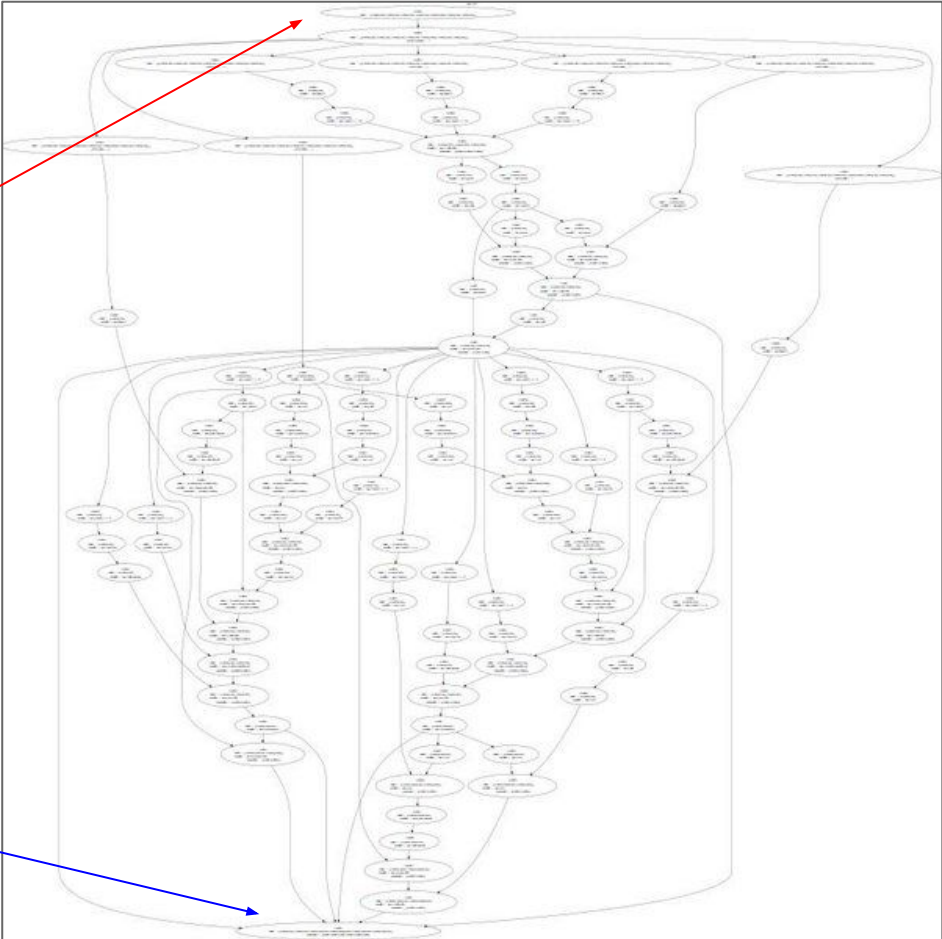
loss



# Neural Turing Machine

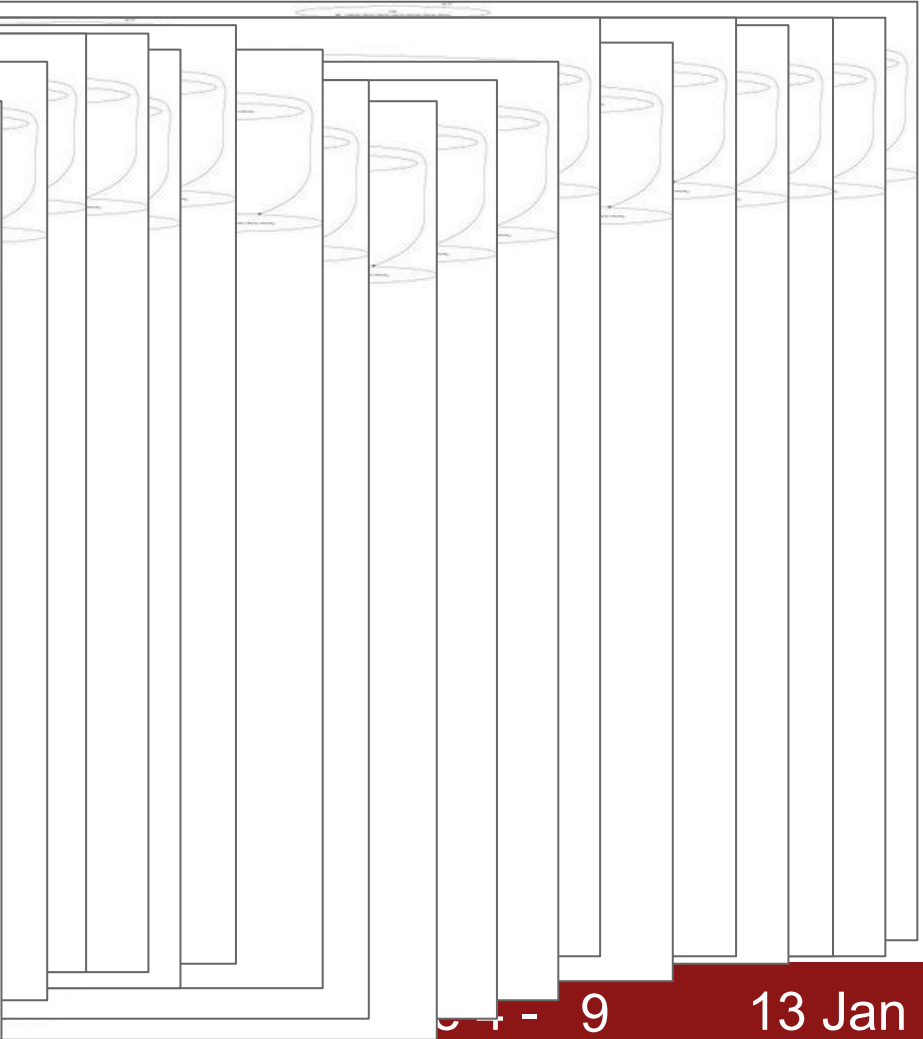
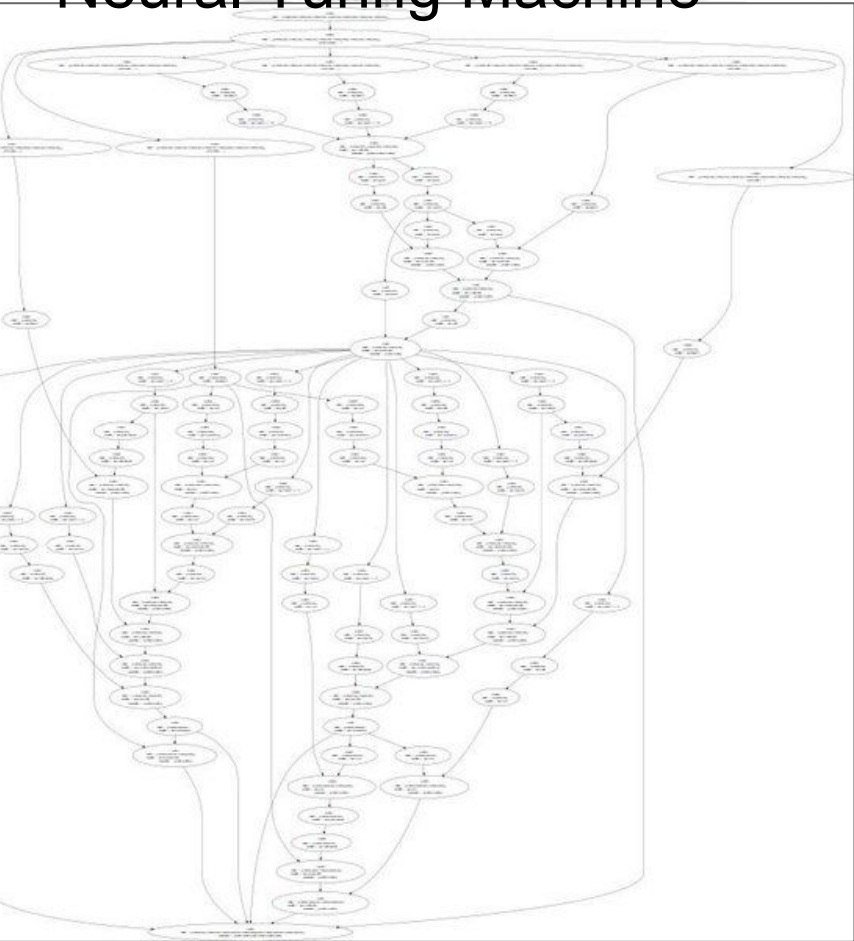
input tape

loss



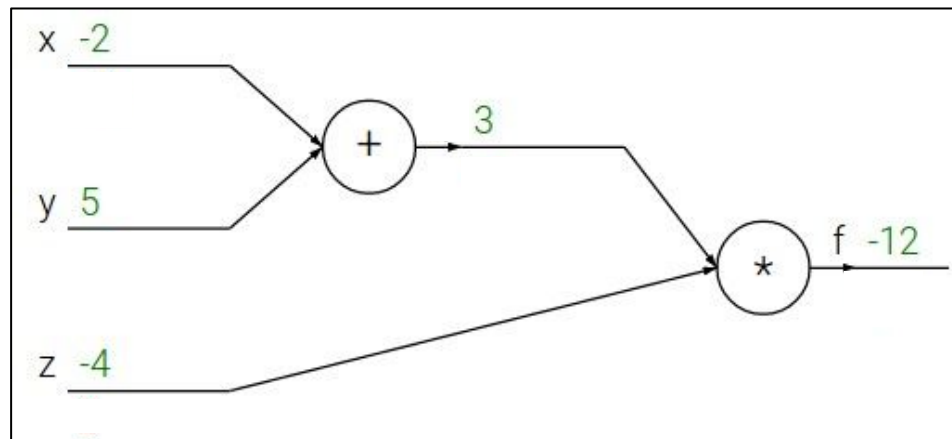


# Neural Turing Machine



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

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



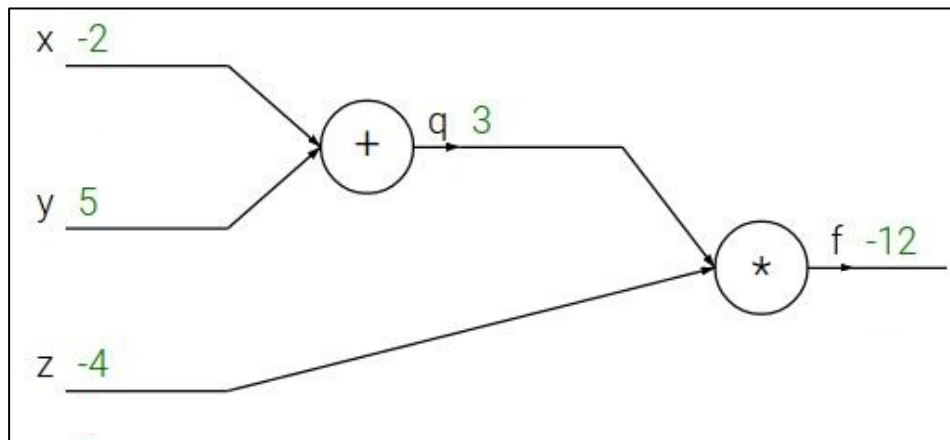
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



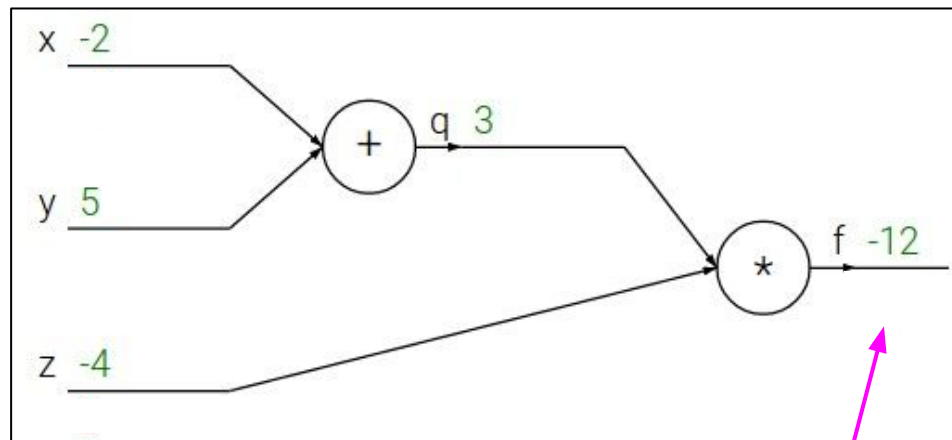
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

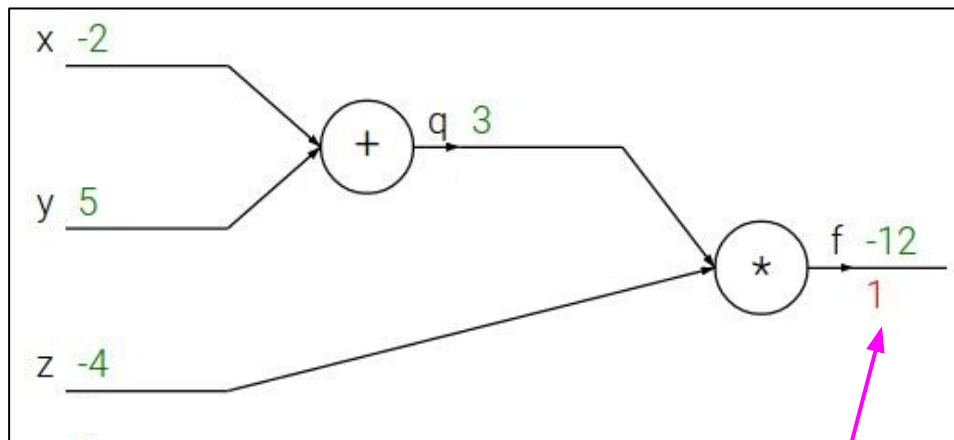
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

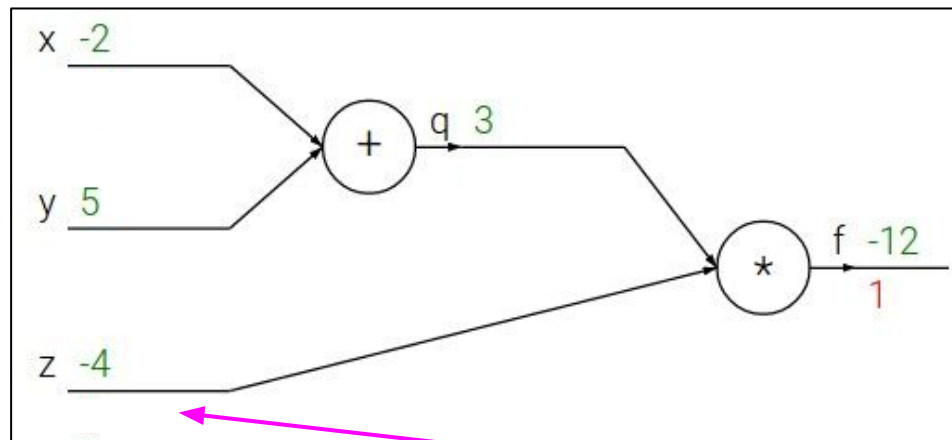
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

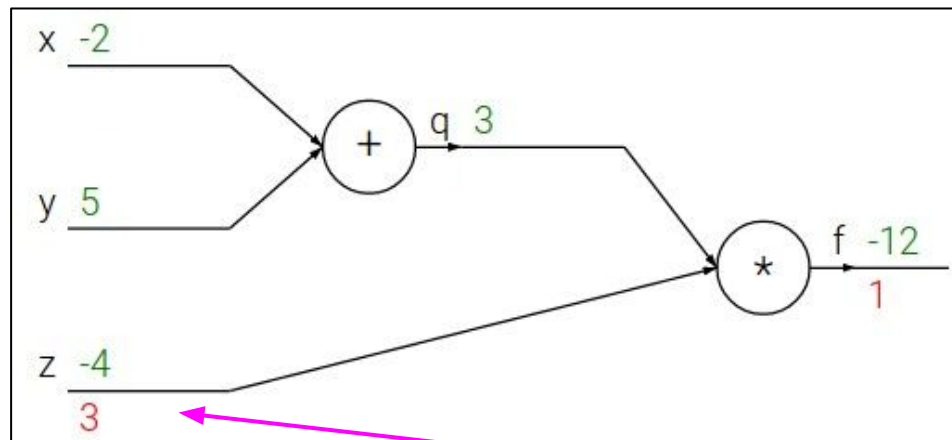
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

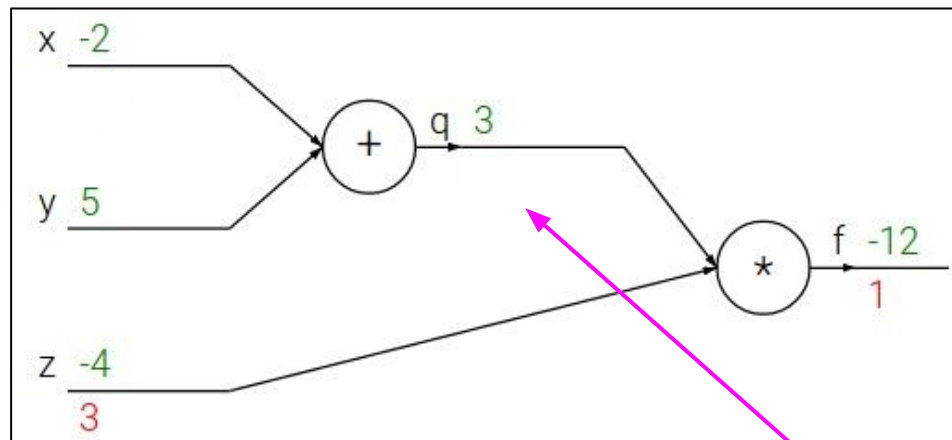
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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



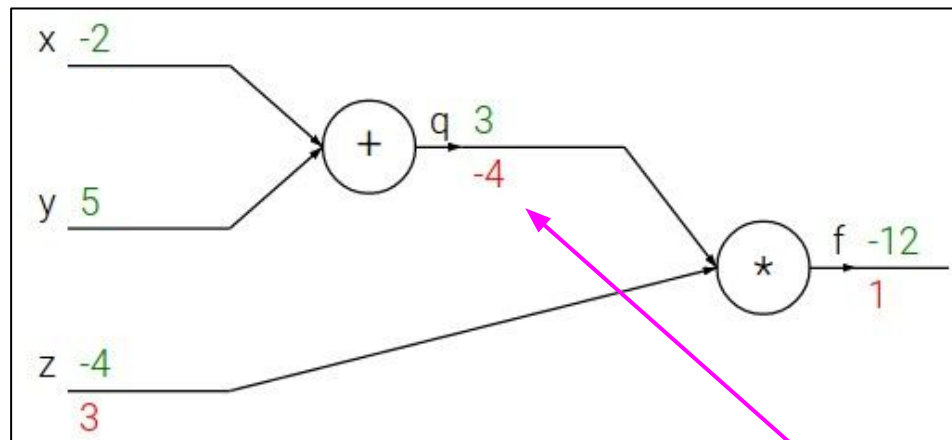
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

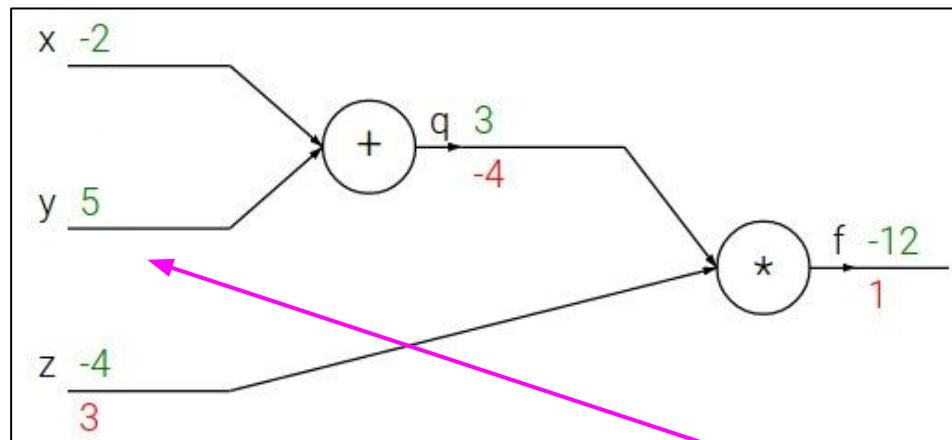
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

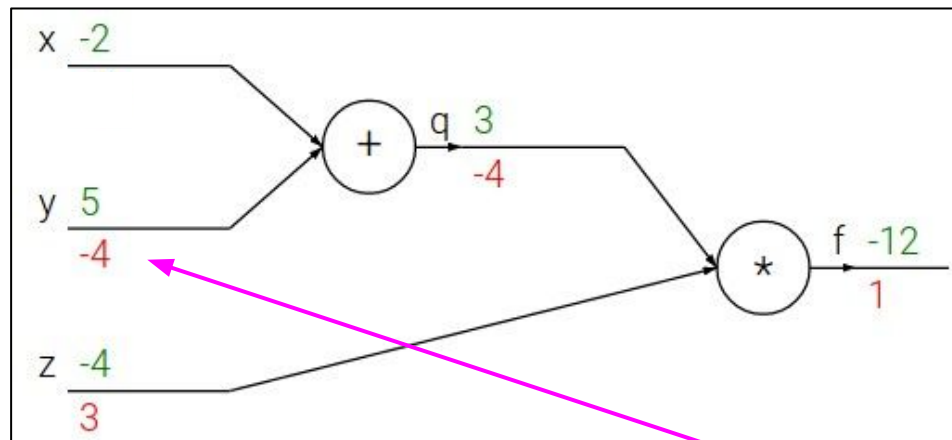
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

Chain rule:

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

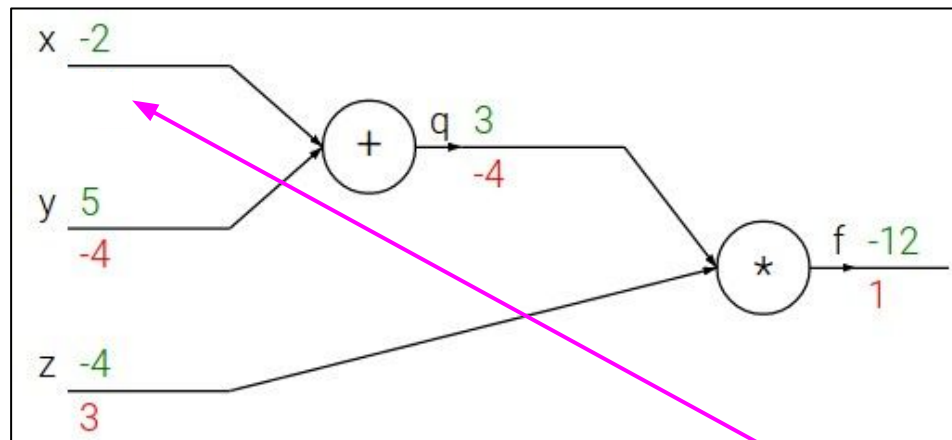
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

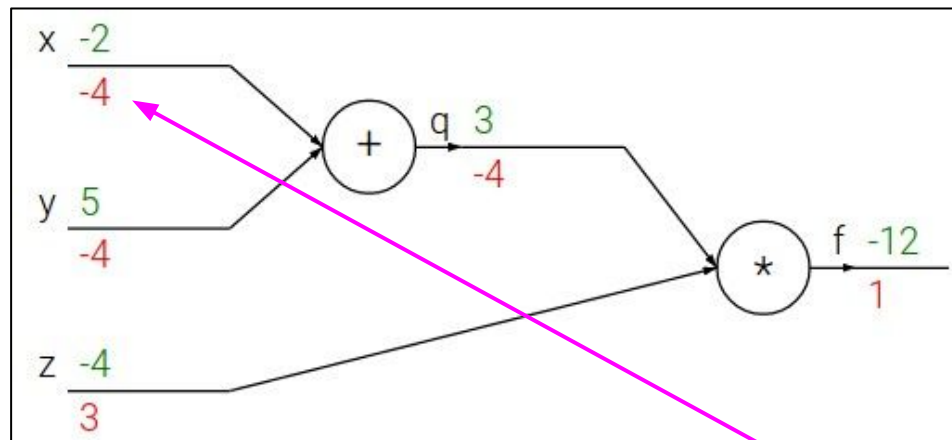
$$f(x, y, z) = (x + y)z$$

$$\text{e.g. } x = -2, y = 5, z = -4$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

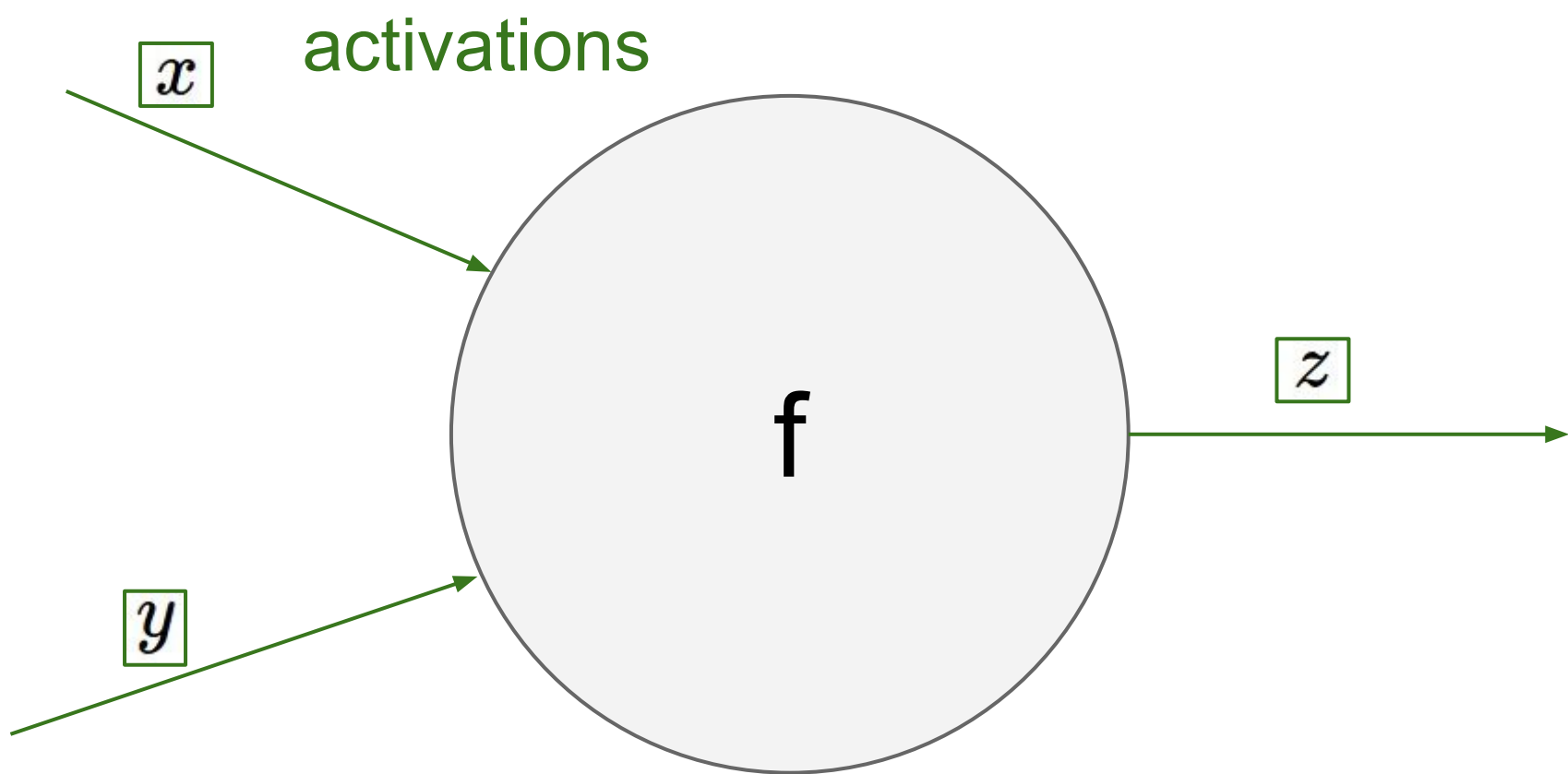
$$\text{Want: } \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



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

Chain rule:

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



activations

$x$

“local gradient”

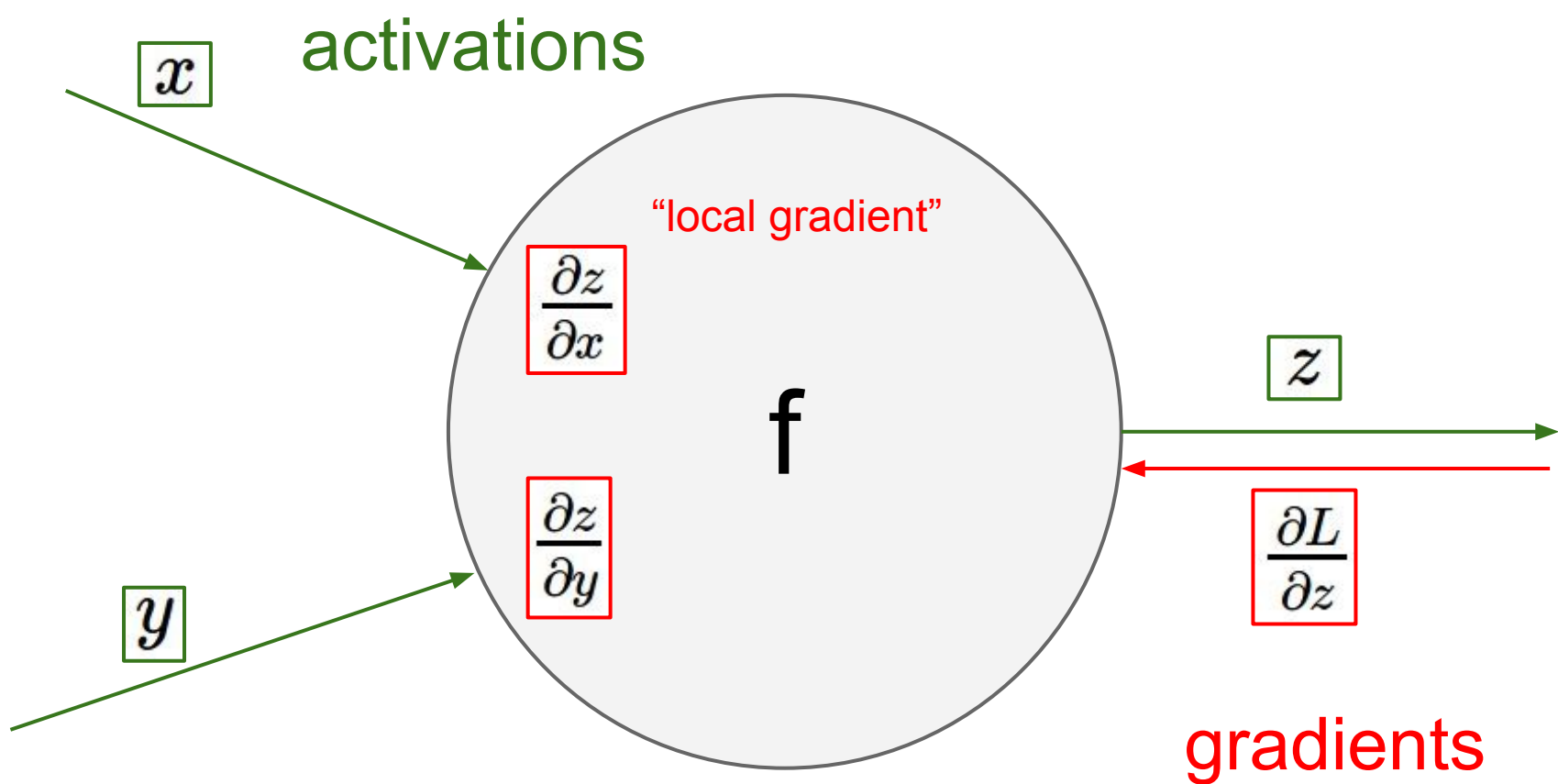
$$\frac{\partial z}{\partial x}$$

$f$

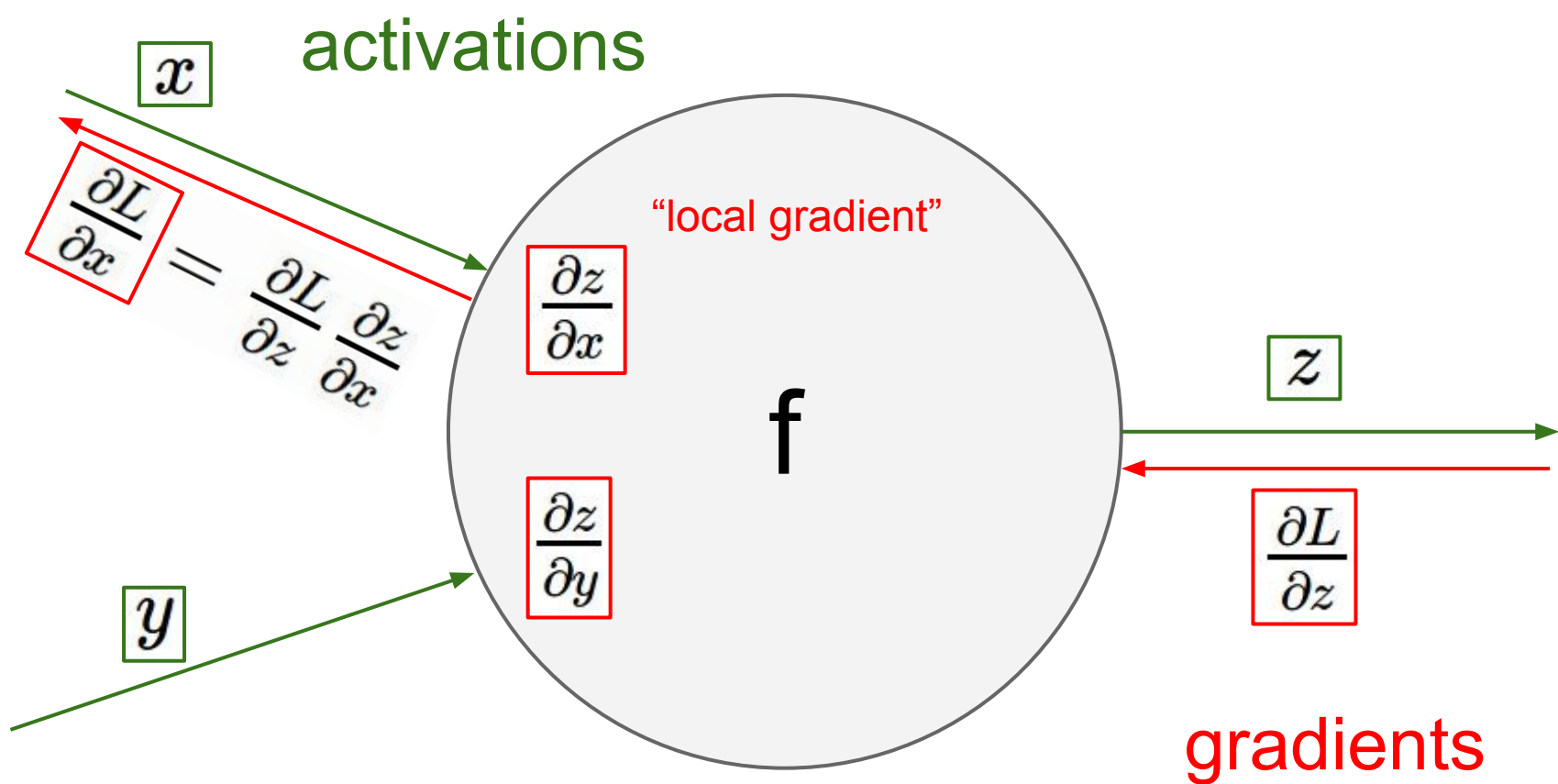
$$\frac{\partial z}{\partial y}$$

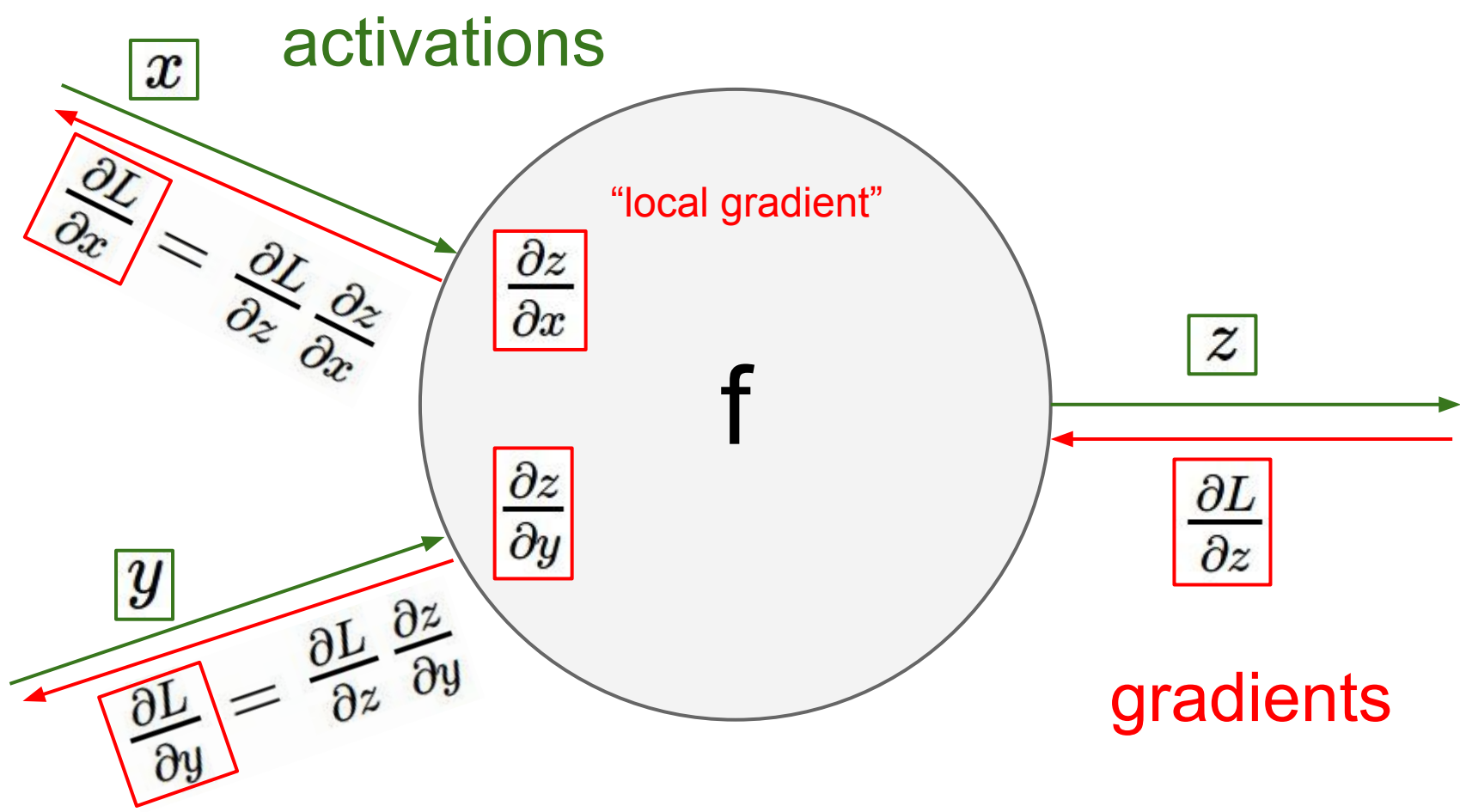
$y$

$z$









activations

$x$

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

"local gradient"

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

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

$f$

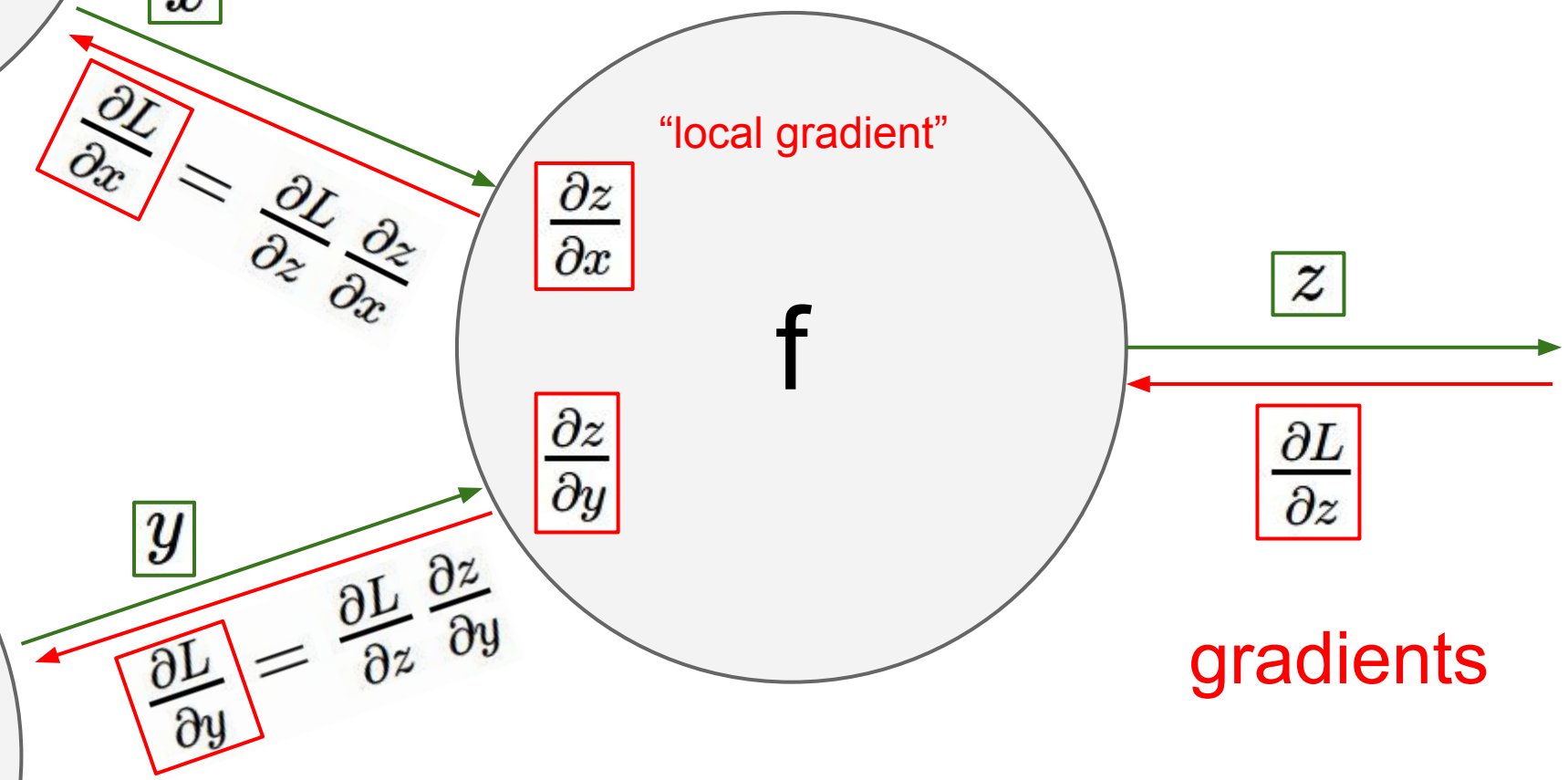
$z$

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

gradients

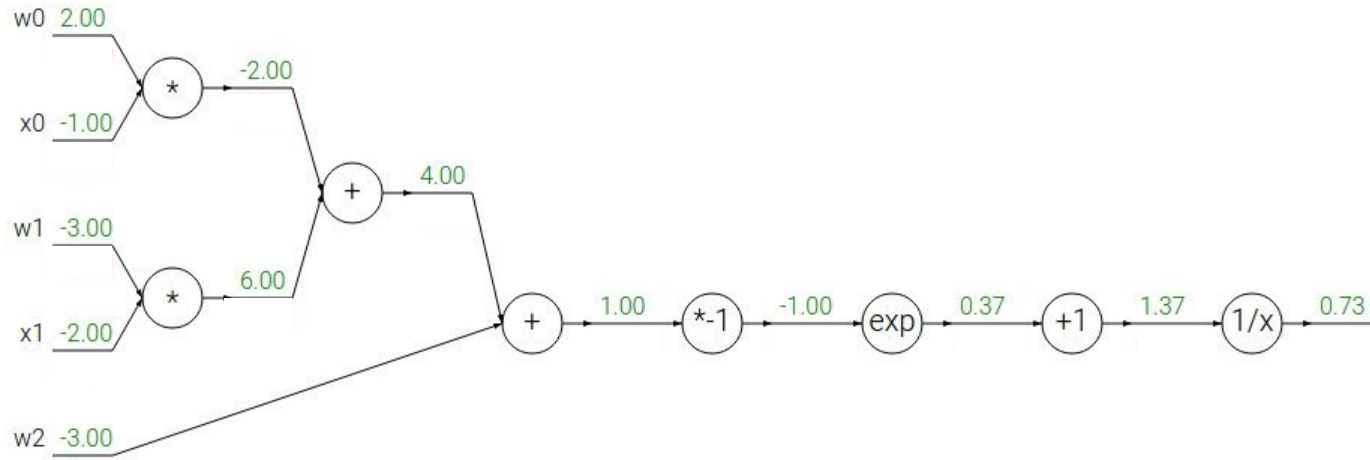
$y$

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



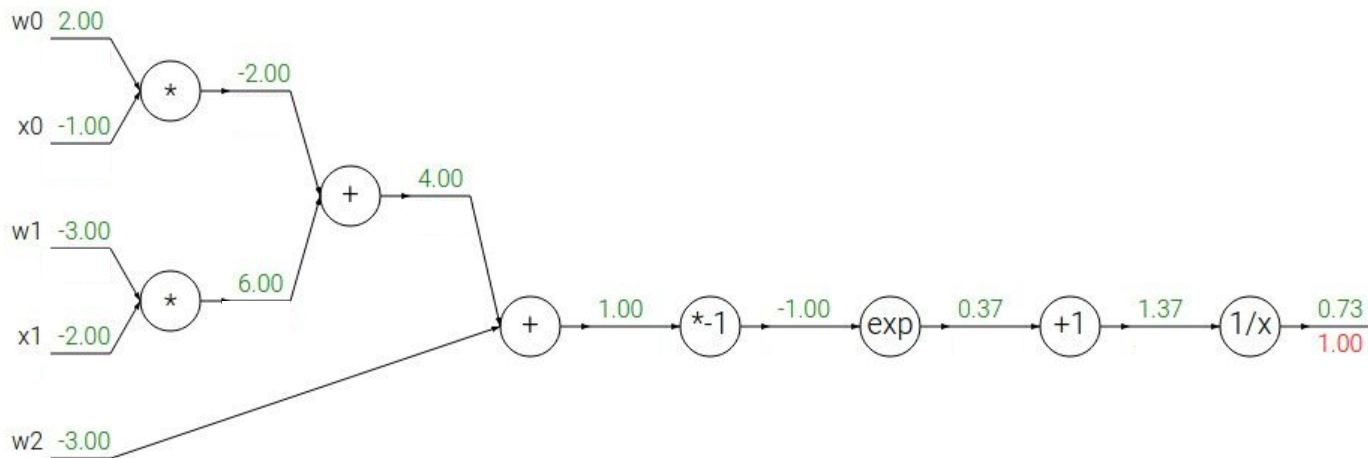
Another example:

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



Another example:

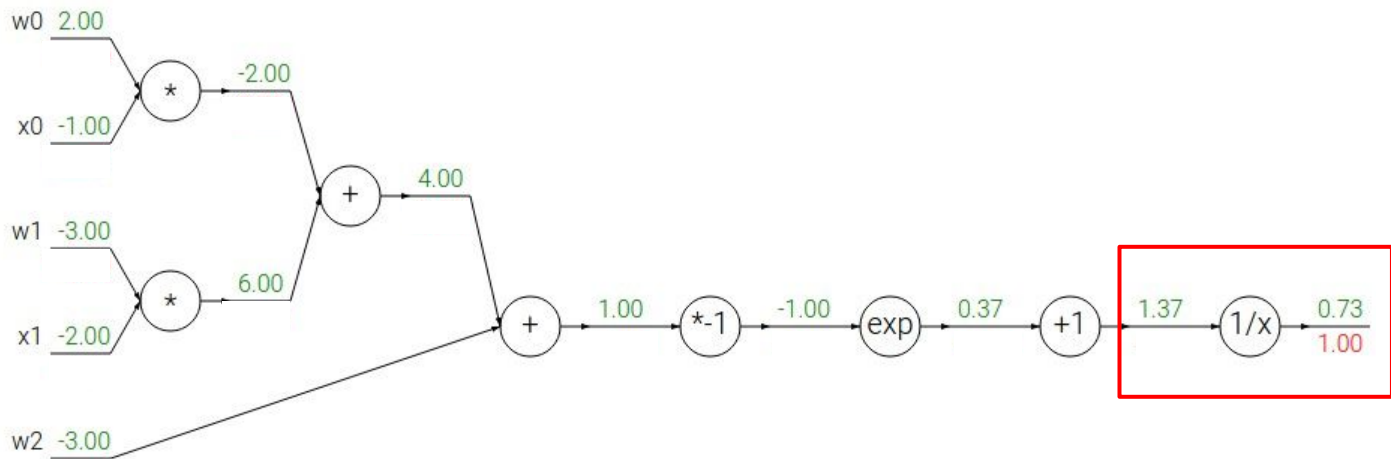
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$f(x) = e^x$	→	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	→	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	→	$\frac{df}{dx} = a$		$f_c(x) = c + x$	→	$\frac{df}{dx} = 1$

Another example:

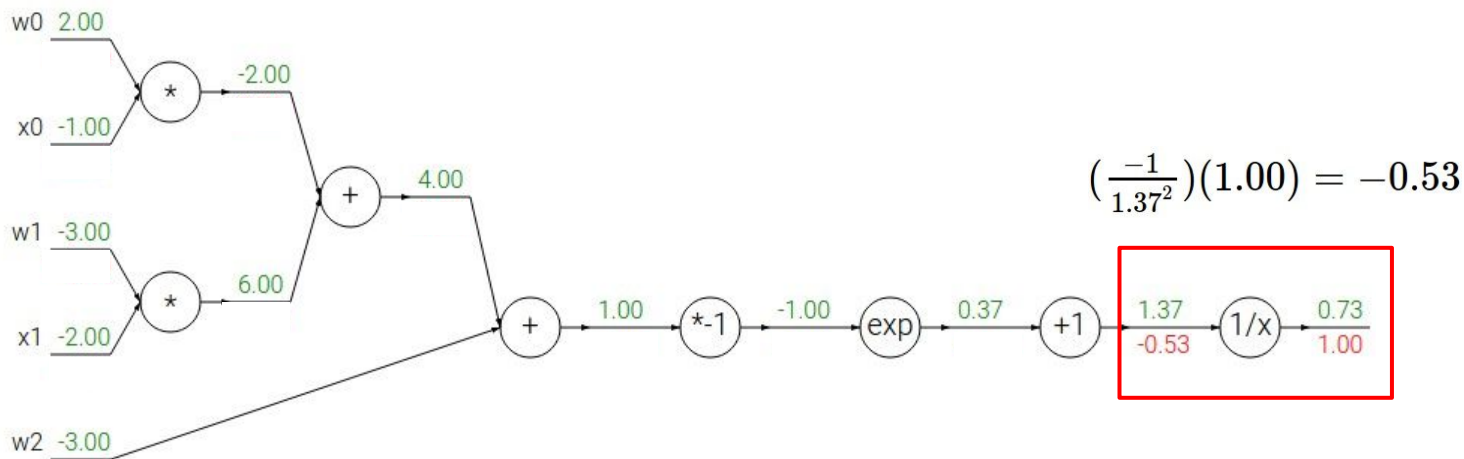
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$		$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

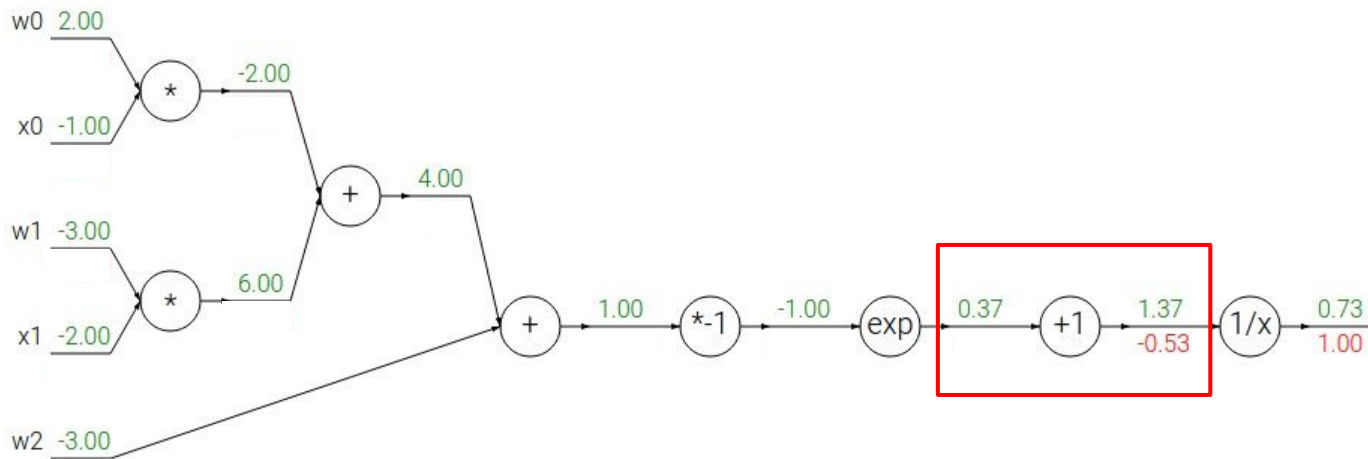
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

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

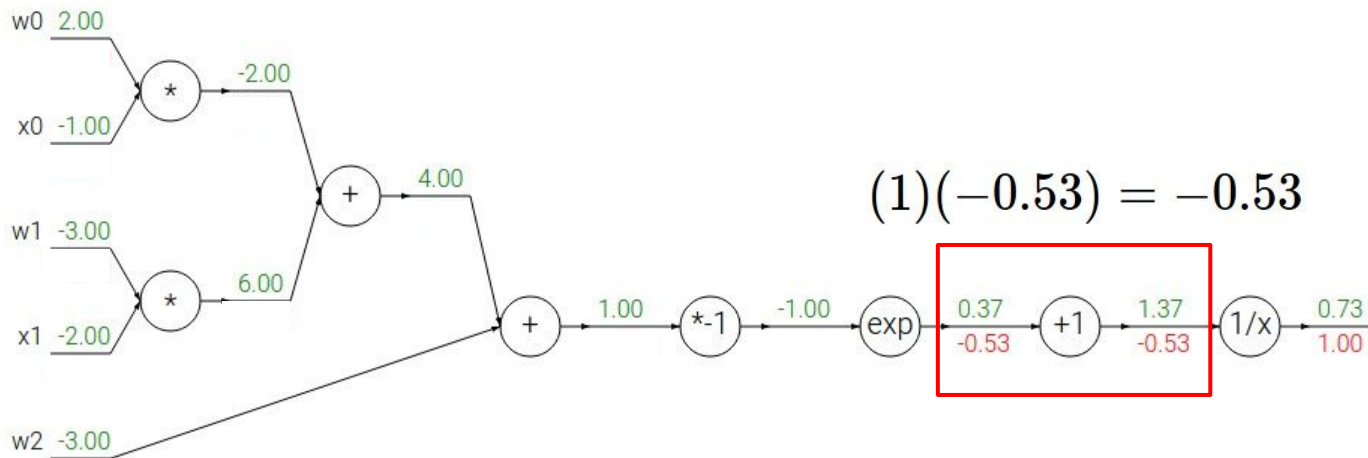


$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$		$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$



Another example:

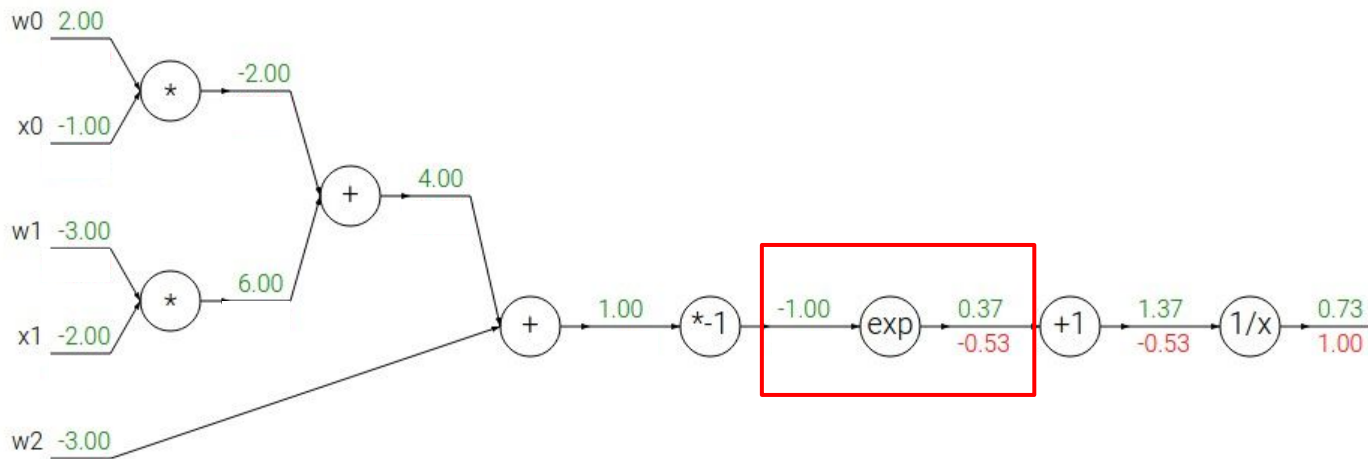
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$		$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

Another example:

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



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

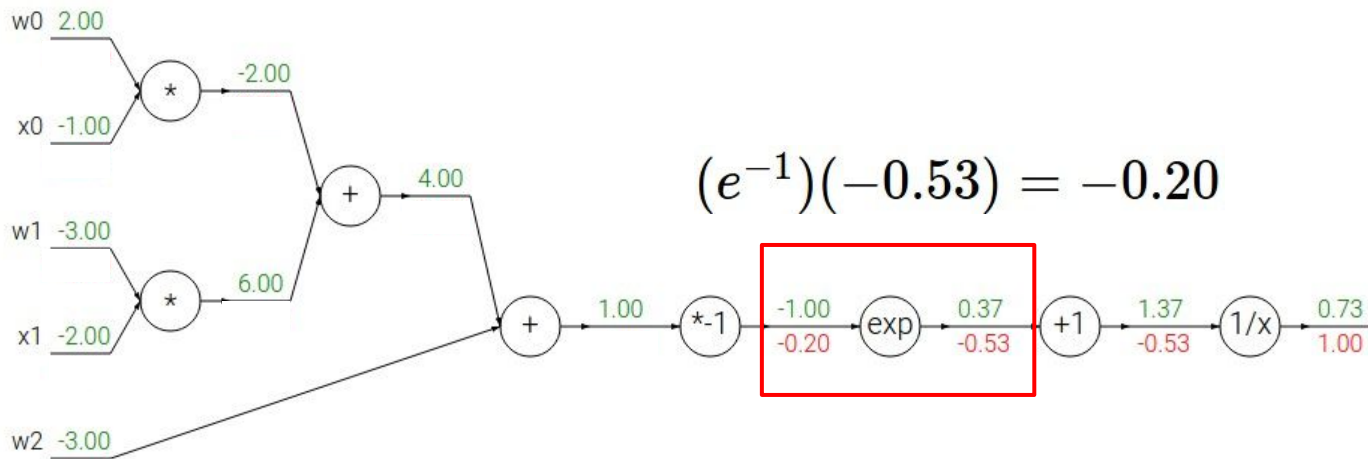
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example:

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



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

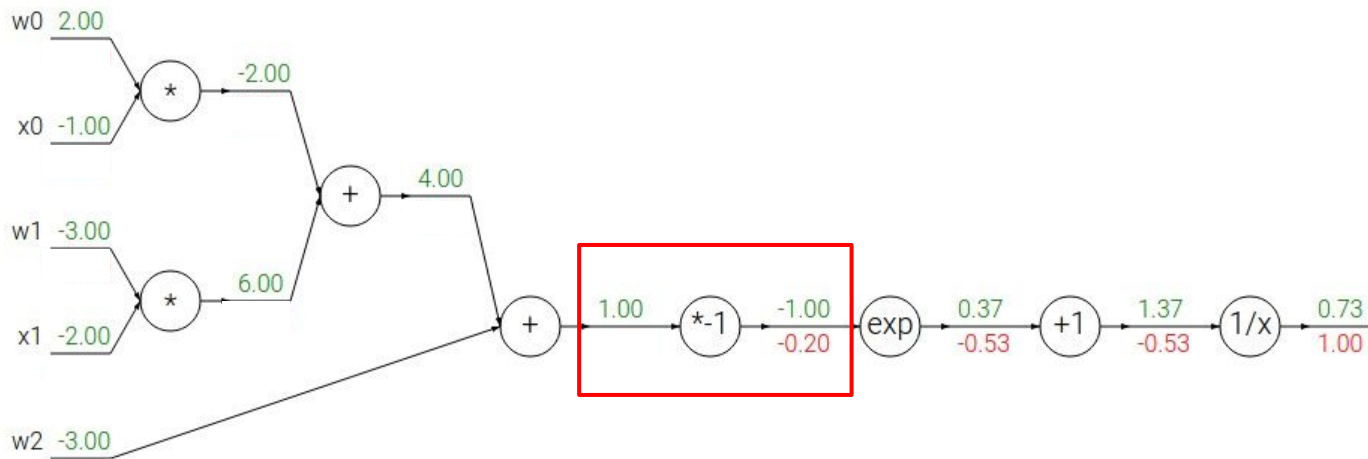
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example:

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

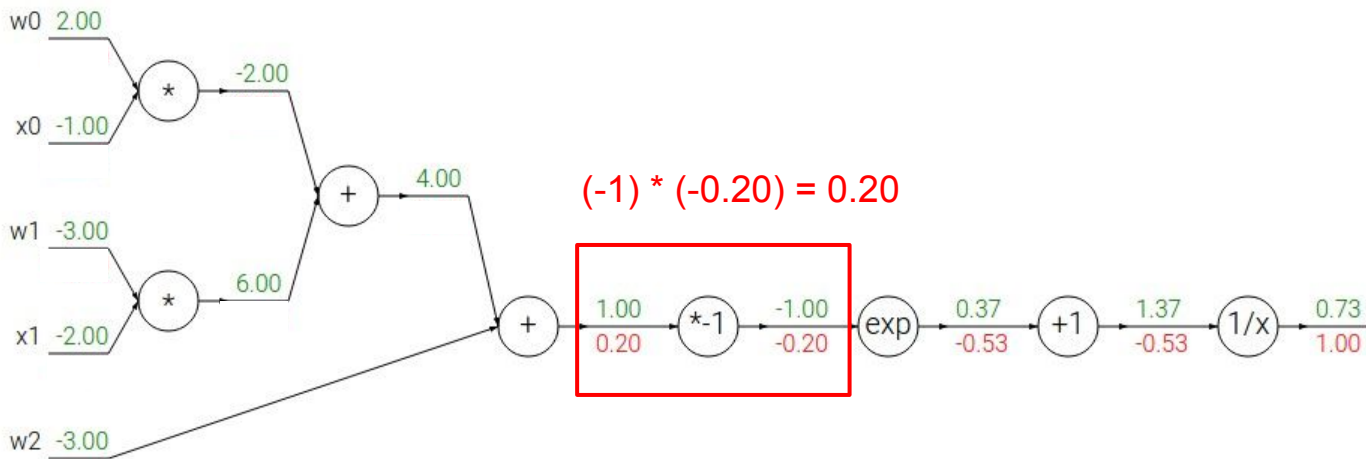
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

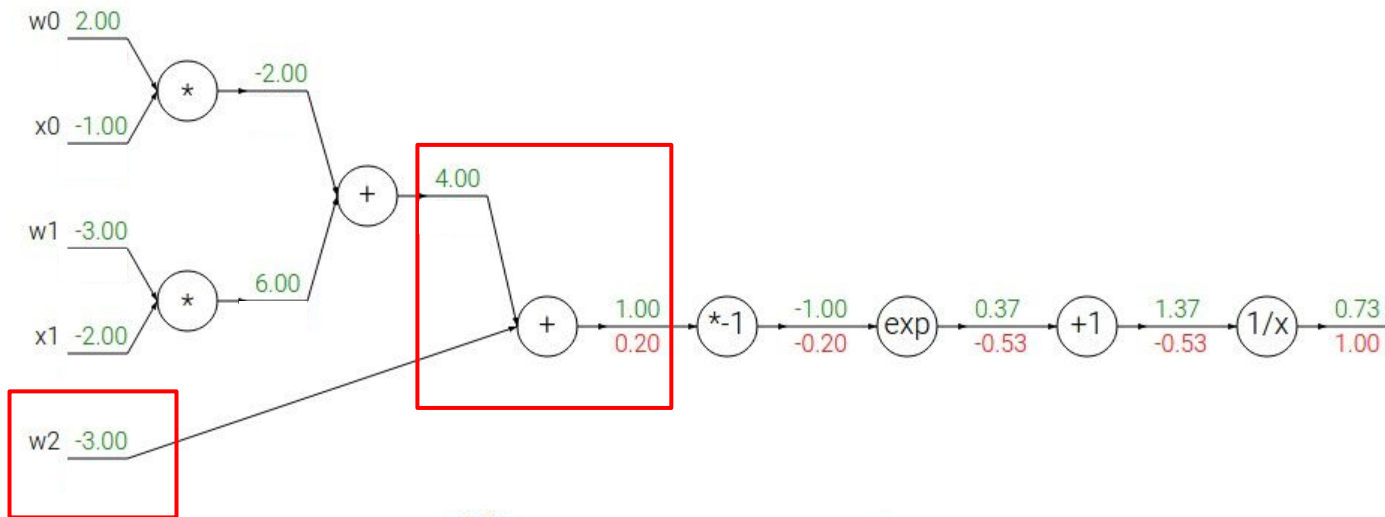
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

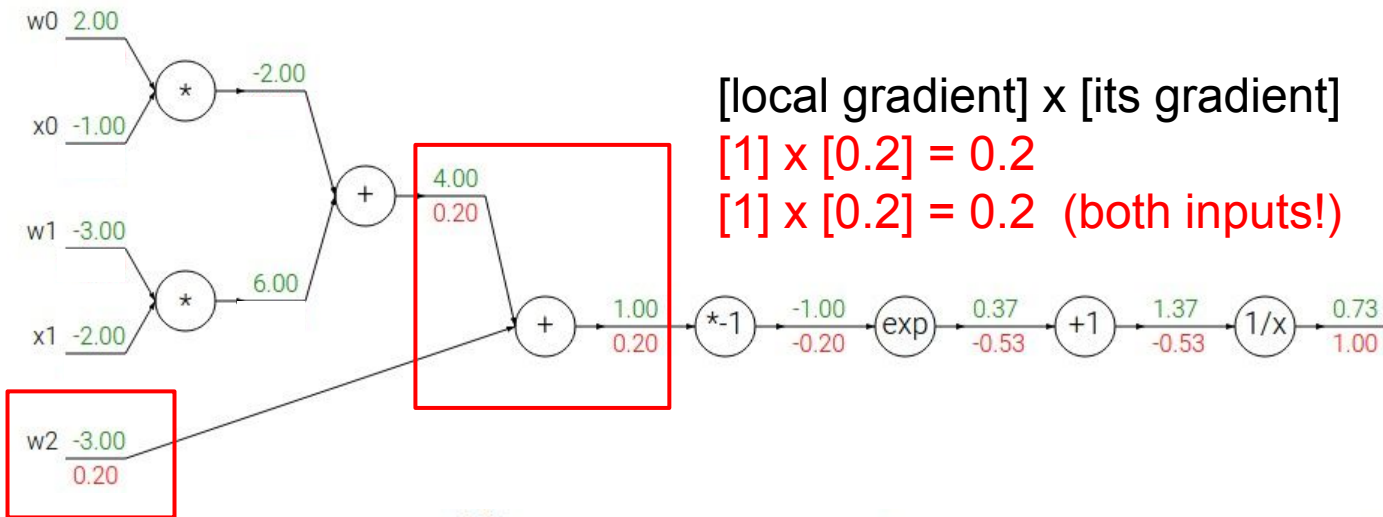
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

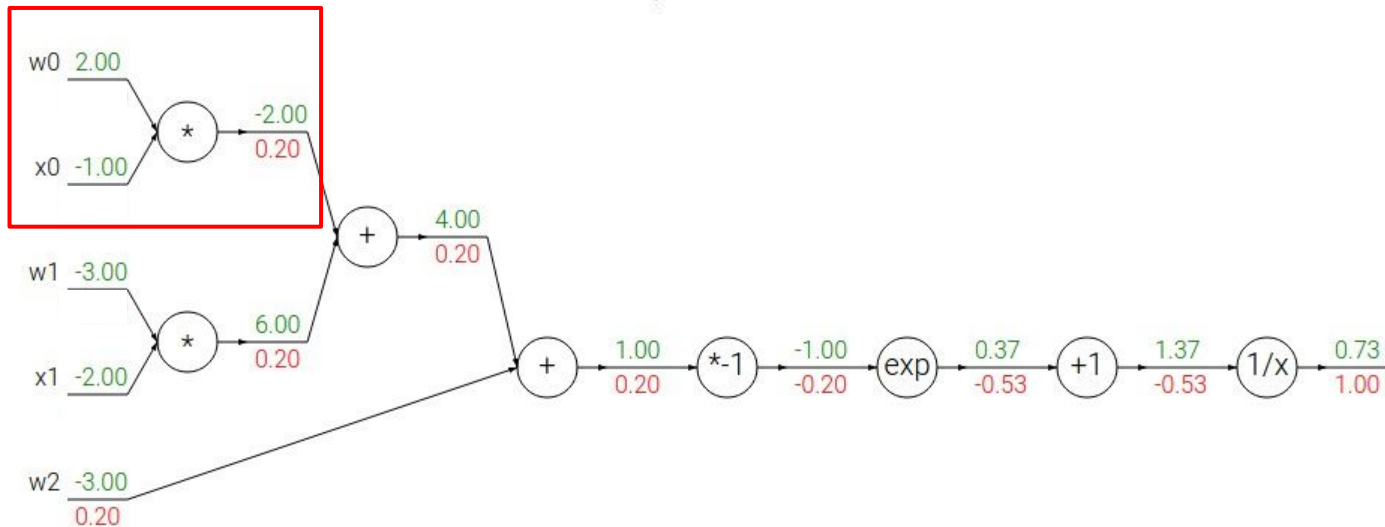
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

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

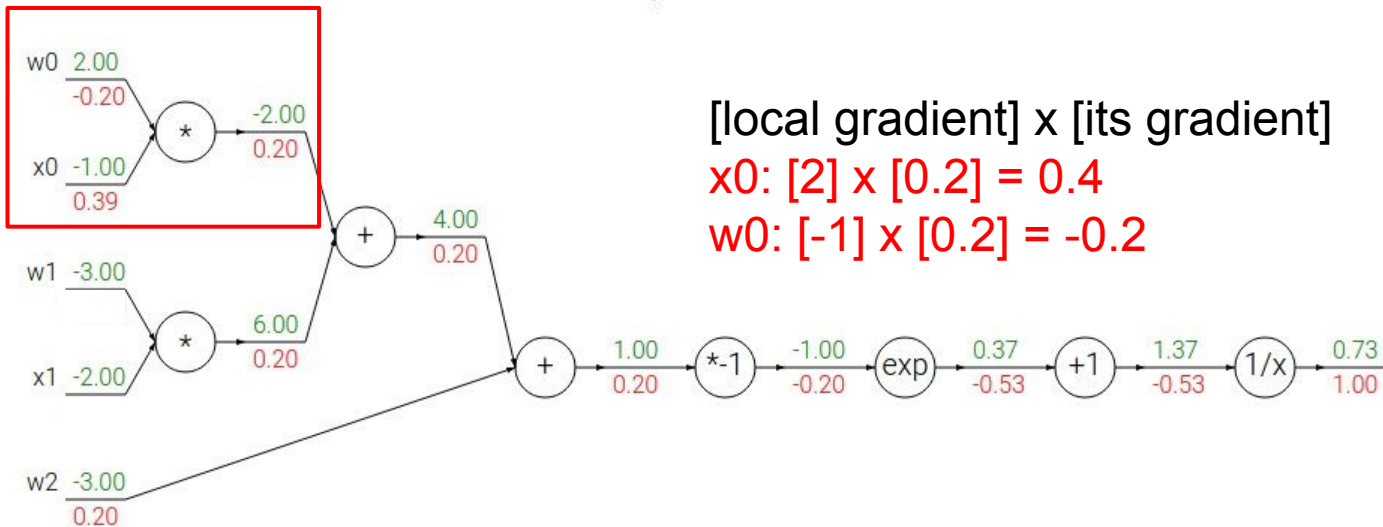


$f(x) = e^x$	→	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	→	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	→	$\frac{df}{dx} = a$		$f_c(x) = c + x$	→	$\frac{df}{dx} = 1$



Another example:

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



[local gradient] x [its gradient]

$x_0: [2] \times [0.2] = 0.4$

$w_0: [-1] \times [0.2] = -0.2$

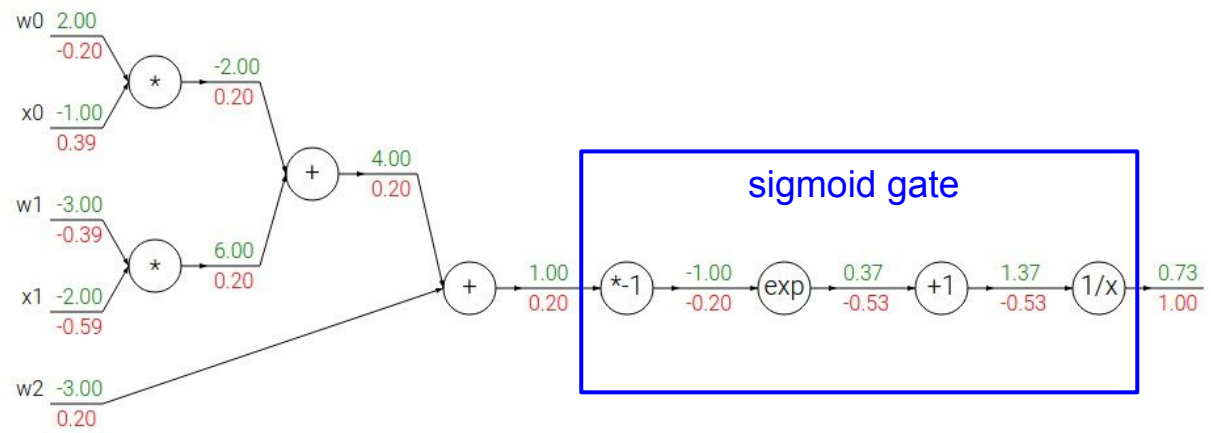
$f(x) = e^x$	→	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	→	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	→	$\frac{df}{dx} = a$		$f_c(x) = c + x$	→	$\frac{df}{dx} = 1$

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$

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

sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

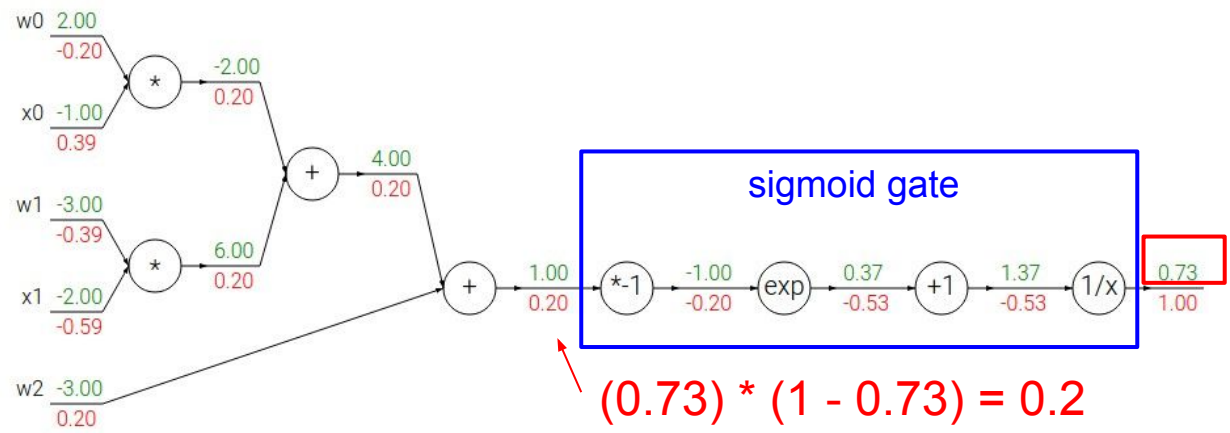


$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$

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

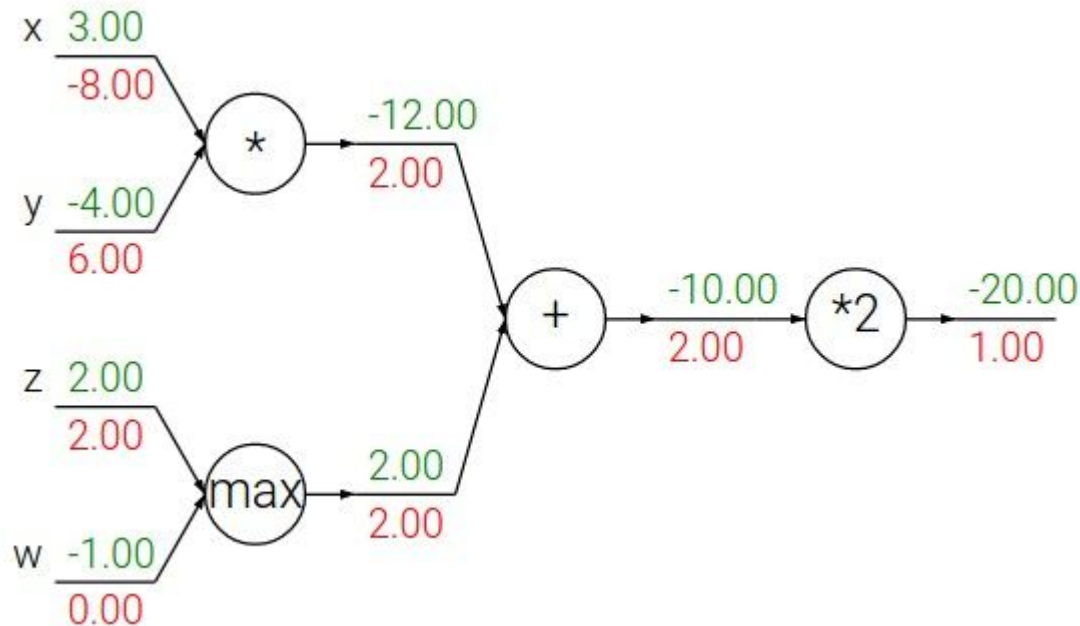
sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}}\right) \left(\frac{1}{1 + e^{-x}}\right) = (1 - \sigma(x))\sigma(x)$$

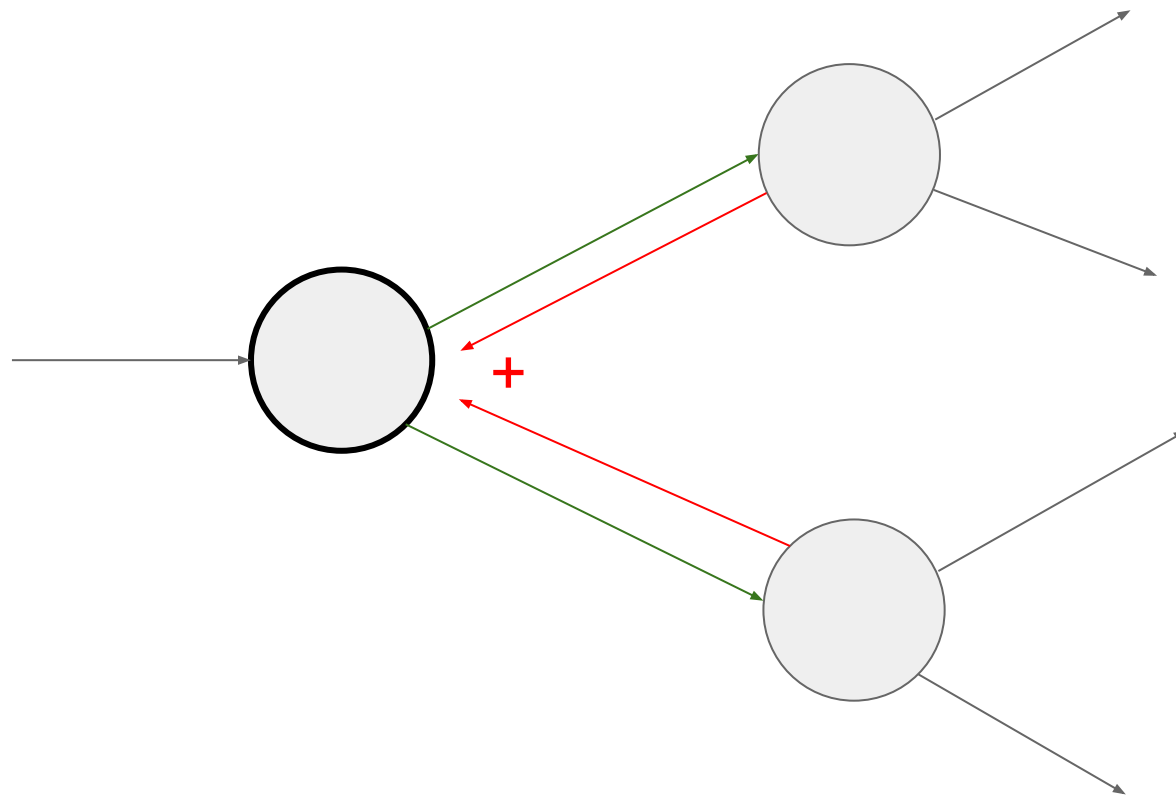


# Patterns in backward flow

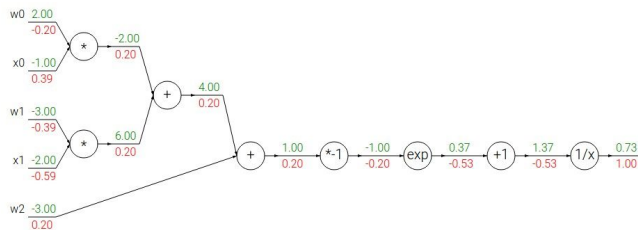
**add gate:** gradient distributor  
**max gate:** gradient router  
**mul gate:** gradient... “switcher”?



# Gradients add at branches



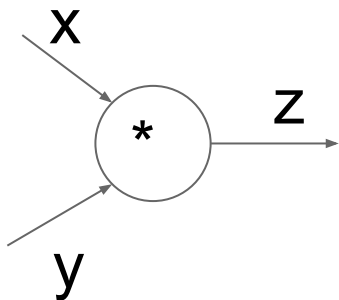
# Implementation: forward/backward API



Graph (or Net) object. (*Rough psuedo code*)

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

# Implementation: forward/backward API

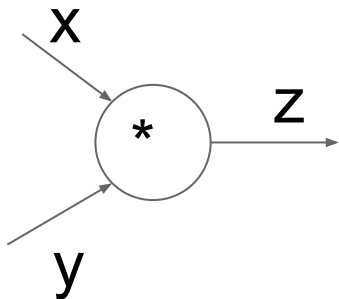


(x,y,z are scalars)

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

Diagram illustrating the forward and backward passes for a MultiplyGate operation. The forward pass (def forward(x,y)) calculates the output z = x\*y. The backward pass (def backward(dz)) calculates the gradients dx and dy. The diagram shows the backward pass receiving the gradient dz and returning [dx, dy]. Arrows point from the  $\frac{\partial L}{\partial z}$  box to the dz parameter in the backward function, and from the  $\frac{\partial L}{\partial x}$  box to the dx element in the return list.

# Implementation: forward/backward API



```
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]
```

(x,y,z are scalars)





# Example: Torch Layers

Code	Watch	Star	Fork
No description or website provided.			
1,639 commits 97 branches 6 releases 11 contributors			
Search	master	New and recent	Download ZIP
📄 <b>sewerh</b> Merge pull request #363 from bartholomewSD/master	Label created 23A17C 15 hours ago		
📄 <b>dic</b> Fix batch mode in ImageResizingCriterion	4 days ago		
📄 <b>genetic</b> Improve error message in SpatialConvolutionMM	4 days ago		
📄 <b>to</b> THNN: add missing OpenMP include	2 days ago		
📄 <b>node</b> Add 'leaf' dependency	14 days ago		
📄 <b>aligners</b> Merge pull request #343 from input	4 months ago		
📄 <b>batchnorm</b> Don't throw batch size to the top level	1 year ago		
📄 <b>batch_pref</b> small fixes for test path	2 months ago		
📄 <b>Atlas.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago		
📄 <b>BatchCriterion.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago		
📄 <b>BatchLU</b> Fix Add with multi batch	10 months ago		
📄 <b>BatchNormalizer</b> Adding inplace AddConstant and MulConstant	9 months ago		
📄 <b>BCEDCriterion.lua</b> Remove unnecessary modules from BCEDCriterion	3 months ago		
📄 <b>BatchNormalization.lua</b> fix batchnorm reset	3 months ago		
📄 <b>ClassNLLoss.lua</b> Merge table modules to return correct number of gradients	6 months ago		
📄 <b>CDLTable.lua</b> Merge table modules to return correct number of gradients	6 months ago		
📄 <b>ClassNLLoss</b> Add C implementation of <a href="#">SpatialBatchNormalization</a>	7 days ago		
📄 <b>CDML.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago		
📄 <b>CDMLTable.lua</b> Merge table modules to return correct number of gradients	6 months ago		
📄 <b>CONTRIB/THNN.md</b> added developing file	3 months ago		
📄 <b>COPYRIGHT</b> add copyright file	2 years ago		
📄 <b>ClassNLLoss</b> Merge table modules to return correct number of gradients	6 months ago		
📄 <b>Clamp.lua</b> Use custom range in HardTanh and mask it in Clamp	3 months ago		
📄 <b>ClassNLLCriterion.lua</b> Add functional conversion of <a href="#">ClassNLLCriterion</a>	13 days ago		
📄 <b>Convex.lua</b> fix bug in conditional expression	4 months ago		
📄 <b>Conv2dTable.lua</b> Merge bug in Conv2dTable variable length	4 months ago		
📄 <b>Convex.lua</b> Adding applyToModule to no Converter, which is like apply() but...	3 months ago		
📄 <b>Copy.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago		
📄 <b>Coarse.lua</b> Fix types in Coarse	4 months ago		
📄 <b>CoarseDistance.lua</b> Do not change state variables in CoarseDistanceConvEmbeddingCriterion	2 months ago		
📄 <b>CoarseEmbeddingCriterion.lua</b> Do not change state variables in CoarseEmbeddingConvEmbeddingCriterion	2 months ago		
📄 <b>Criterion.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago		
📄 <b>CriterionTable.lua</b> Remove unpatch to table unpatch for Lua 5.2	8 months ago		
📄 <b>CoarseEmbeddingCriterion.lua</b> Check for 'no Module' and 'no Criterion' in recursiveType	8 months ago		
📄 <b>DeepConvConv.lua</b> adding default backward to Convex, DeepConvConv, Sequential	9 months ago		
📄 <b>ClassNLLCriterion.lua</b> Use user's THNN functions over the default nearest outputs	10 days ago		
📄 <b>DotProductLoss</b> Add batch mode in DotProductLoss unit test	2 months ago		
📄 <b>Dropout.lua</b> in-place dropout	4 months ago		
📄 <b>ELU.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago		
📄 <b>ErrorHandler.lua</b> Give better error messages when trying to use the wrong kind of Tensor	1 year ago		
📄 <b>Embedder.lua</b> no Module presave type sharing semantics (#187), add no Module apply	6 months ago		
📄 <b>Exp.lua</b> Exp made lua only	9 months ago		
📄 <b>FilterTable.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago		
📄 <b>GradientDescent.lua</b> Add GradientDescent layer	4 months ago		
📄 <b>HardTanh.lua</b> Add functional conversion of <a href="#">HardTanh</a>	10 days ago		
📄 <b>HardTanhTable.lua</b> Add functional conversion of <a href="#">HardTanh</a>	10 days ago		
📄 <b>ImageResizingCriterion.lua</b> rewrite ImageResizingCriterion to support batch mode	6 months ago		
📄 <b>Identify.lua</b> Revert to previous identify lua implementation	2 months ago		
📄 <b>Index.lua</b> Simplifying and more efficient no Index	2 months ago		
📄 <b>Isaacson.lua</b> Add unit tests for <a href="#">hennon.lua</a> , fix bugs detected by the tests	6 months ago		
📄 <b>LeafTable.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago		
📄 <b>L1Cost.lua</b> Use tensor for THNN functions over for single element outputs	10 days ago		
📄 <b>L1HingeEmbeddingCriterion.lua</b> Make types() fully recursive	9 months ago		
📄 <b>L1Penalty.lua</b> fixed L1Penalty constructor arguments	1 year ago		
📄 <b>LeakyReLU.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago		
📄 <b>Linear.lua</b> Remove spurious modules from no Linear	4 months ago		

📄 <b>LogSoftMax.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago
📄 <b>LogSoftMaxTable.lua</b> Add THNN conversion of <a href="#">ELU</a> , <a href="#">LeakyReLU</a> , <a href="#">LogSigmoid</a> , <a href="#">LogSoftMax</a> , <a href="#">Looku...</a>	7 days ago
📄 <b>LookupTable.lua</b> Harmonize LookupTable signature with some input	5 days ago
📄 <b>MM.lua</b> Remove unpatch to table unpatch for Lua 5.2	8 months ago
📄 <b>MLCriterion.lua</b> Add SoftMargin loss to criterion in the constructor	2 months ago
📄 <b>ImageResizingCriterion.lua</b> implement ImageResizingCriterion	1 year ago
📄 <b>MergeBatchingCriterion.lua</b> Fix batch mode in MergeBatchingCriterion	4 days ago
📄 <b>Misc.lua</b> Merge pull request #404 from rignathator	2 months ago
📄 <b>Misc.lua</b> Add support for negative dimension and both batch and non batch input...	2 months ago
📄 <b>Misc.lua</b> Merge pull request #404 from rignathator	2 months ago
📄 <b>Min.lua</b> correct unpatch variables and custom expressions	20 days ago
📄 <b>Module.lua</b> Revert "Don't use <code>batch</code> parameter, if they are already defined"	10 hours ago
📄 <b>Mul.lua</b> removing the requirement for providing size in no Mul	1 year ago
📄 <b>MulConstant.lua</b> ignore updateGradient if self:gradient() is nil	3 months ago
📄 <b>MLCriterion.lua</b> asserts in <a href="#">MLCriterion</a> and <a href="#">ParallelCriterion</a> unit	2 months ago
📄 <b>MultibatchCriterion.lua</b> main source of torch7 test	1 year ago
📄 <b>MultMergeCriterion.lua</b> multimerge supports p=2	11 months ago
📄 <b>Narrow.lua</b> Spoke in <code>Narrow</code> not done in place.	6 months ago
📄 <b>NormTable.lua</b> NormTable	6 months ago
📄 <b>NormTable.lua</b> Remove term and softmax from NormTable, because they allocate memory...	20 days ago
📄 <b>ORMLU.lua</b> Buffer for ORMLU unit implementation	8 months ago
📄 <b>Padding.lua</b> fixed broken no Padding: input was returned in backward	5 months ago
📄 <b>ParallelCriterion.lua</b> Merge pull request #432 from ewergenerator	20 days ago
📄 <b>Parallel.lua</b> fix bug in conditional expression	1 month ago
📄 <b>ParallelCriterion.lua</b> asserts in <a href="#">ParallelCriterion</a> and <a href="#">ParallelCriterion</a> unit	2 months ago
📄 <b>ParallelTable.lua</b> module conversion: ParallelTable inherits Criterion, unit tests	1 year ago
📄 <b>Power.lua</b> Use <code>UNCL</code> test endings	7 months ago
📄 <b>README.md</b> doc: README	5 months ago
📄 <b>ReLU.lua</b> Add optionized <code>relu</code> method (tensor and <code>ReLU</code> )	3 months ago
📄 <b>ReLU.lua</b> adds module <code>ReLU</code> and fixes a potential double-free issue in <code>SoftP...</code>	8 months ago
📄 <b>ReLUTable.lua</b> Replaces batch mode	8 months ago
📄 <b>ReLUTable.lua</b> Added more informative pretty printing	1 year ago
📄 <b>Select.lua</b> initial version of torch7 test	4 years ago
📄 <b>SelectTable.lua</b> no Module presave type sharing semantics (#187), add no Module apply	4 months ago
📄 <b>Sequential.lua</b> Merge Sequential remove inner case	6 months ago
📄 <b>Sign.lua</b> initial version of torch7 test	4 years ago
📄 <b>SmoothL1Criterion.lua</b> Add SoftMargin loss to criterion in the constructor	2 months ago
📄 <b>SoftMax.lua</b> Fix various unused variables in no	1 year ago
📄 <b>SoftMax.lua</b> Fix various unused variables in no	1 year ago
📄 <b>SoftMaxTable.lua</b> Fix various unused variables in the SoftMax module (it breaks for input >...	2 years ago
📄 <b>SoftShrink.lua</b> initial version of torch7 test	4 years ago
📄 <b>SoftSign.lua</b> initial version of torch7 test	4 years ago
📄 <b>SpatialBatchNorm.lua</b> Fix various unused variables in no	1 year ago
📄 <b>SpatialBatchNorm.lua</b> Using tensor implementation of <code>applyToParameters</code> for <code>SpatialBatch...</code>	1 month ago
📄 <b>SpatialBatchNormPooling</b> Add <code>SpatialBatchNormPooling</code>	1 year ago
📄 <b>SpatialAveragePooling.lua</b> SpatialAveragePooling supports padding, cell mode and exclude_grad & r...	20 days ago
📄 <b>SpatialBatchNormalization</b> Add C implementation of <a href="#">SpatialBatchNormalization</a>	7 days ago
📄 <b>SpatialConvolutionNormaliz...</b> Make types() fully recursive	9 months ago
📄 <b>SpatialConvolution.lua</b> Fix types in <a href="#">SpatialConvolution</a>	3 months ago
📄 <b>SpatialConvolutionTable.lua</b> Fix types in <a href="#">SpatialConvolution</a>	3 months ago
📄 <b>SpatialConvolutionMq.lua</b> Remove unpatch and expensive initialization logic from <a href="#">SpatialConv...</a>	8 months ago
📄 <b>SpatialConvolutionMq.lua</b> casts consistency	18 days ago
📄 <b>SpatialConvolutionNormalizati...</b> SpatialConvolutionDivideSubtractiveNormalization work with lat...	8 months ago
📄 <b>SpatialDropout.lua</b> small fix on error message	6 months ago
📄 <b>SpatialFullConvolution</b> Add <code>SpatialFullConvolution</code>	8 months ago
📄 <b>SpatialFullConvolutionMq.lua</b> Add additional terms to <a href="#">SpatialFullConvolution</a> to control the size of ...	5 days ago
📄 <b>SpatialFullConvolutionMq.lua</b> New NN classes	3 years ago
📄 <b>SpatialFullConvolutionMq.lua</b> SpatialAveragePooling divides by <code>kernel*</code>	10 months ago
📄 <b>SpatialFullConvolutionMq.lua</b> SpatialBatchNormPooling supports padding and cell mode	6 months ago
📄 <b>SpatialFullConvolutionMq.lua</b> Add <code>SpatialBatchNormPooling</code>	20 days ago
📄 <b>SpatialFullConvolutionMq.lua</b> Update <code>SoftMax</code> to work in <code>padding</code> mode	4 months ago
📄 <b>SpatialFullConvolutionMq.lua</b> Merge branch 'no_test_new'	3 years ago
📄 <b>SpatialSubtractiveNormalizati...</b> SpatialConvolutionDivideSubtractiveNormalization work with lat...	8 months ago
📄 <b>SpatialSubtractiveNormalizati...</b> Use <code>UNCL</code> test endings	7 months ago
📄 <b>SpatialSubtractiveNormalizati...</b> Added more informative pretty printing	1 year ago
📄 <b>SpatialTable.lua</b> Add support for negative indices in no <code>SoftTable</code>	7 months ago



# Example: Torch MulConstant

$$f(X) = aX$$

```
1 local MulConstant, parent = torch.class('nn.MulConstant', 'nn.Module')
2
3 function MulConstant:__init(constant_scalar, ip)
4     parent.__init(self)
5     assert(type(constant_scalar) == 'number', 'input is not scalar!')
6     self.constant_scalar = constant_scalar
7
8     -- default for inplace is false
9     self.inplace = ip or false
10    if (ip and type(ip) ~= 'boolean') then
11        error('in-place flag must be boolean')
12    end
13 end
```

initialization

```
15 function MulConstant:updateOutput(input)
16     if self.inplace then
17         input:mul(self.constant_scalar)
18         self.output = input
19     else
20         self.output:resizeAs(input)
21         self.output:copy(input)
22         self.output:mul(self.constant_scalar)
23     end
24     return self.output
25 end
```

forward()

```
27 function MulConstant:updateGradInput(input, gradOutput)
28     if self.gradInput then
29         if self.inplace then
30             gradOutput:mul(self.constant_scalar)
31             self.gradInput = gradOutput
32             -- restore previous input value
33             input:div(self.constant_scalar)
34         else
35             self.gradInput:resizeAs(gradOutput)
36             self.gradInput:copy(gradOutput)
37             self.gradInput:mul(self.constant_scalar)
38         end
39         return self.gradInput
40     end
41 end
```

backward()



# Caffe Sigmoid Layer

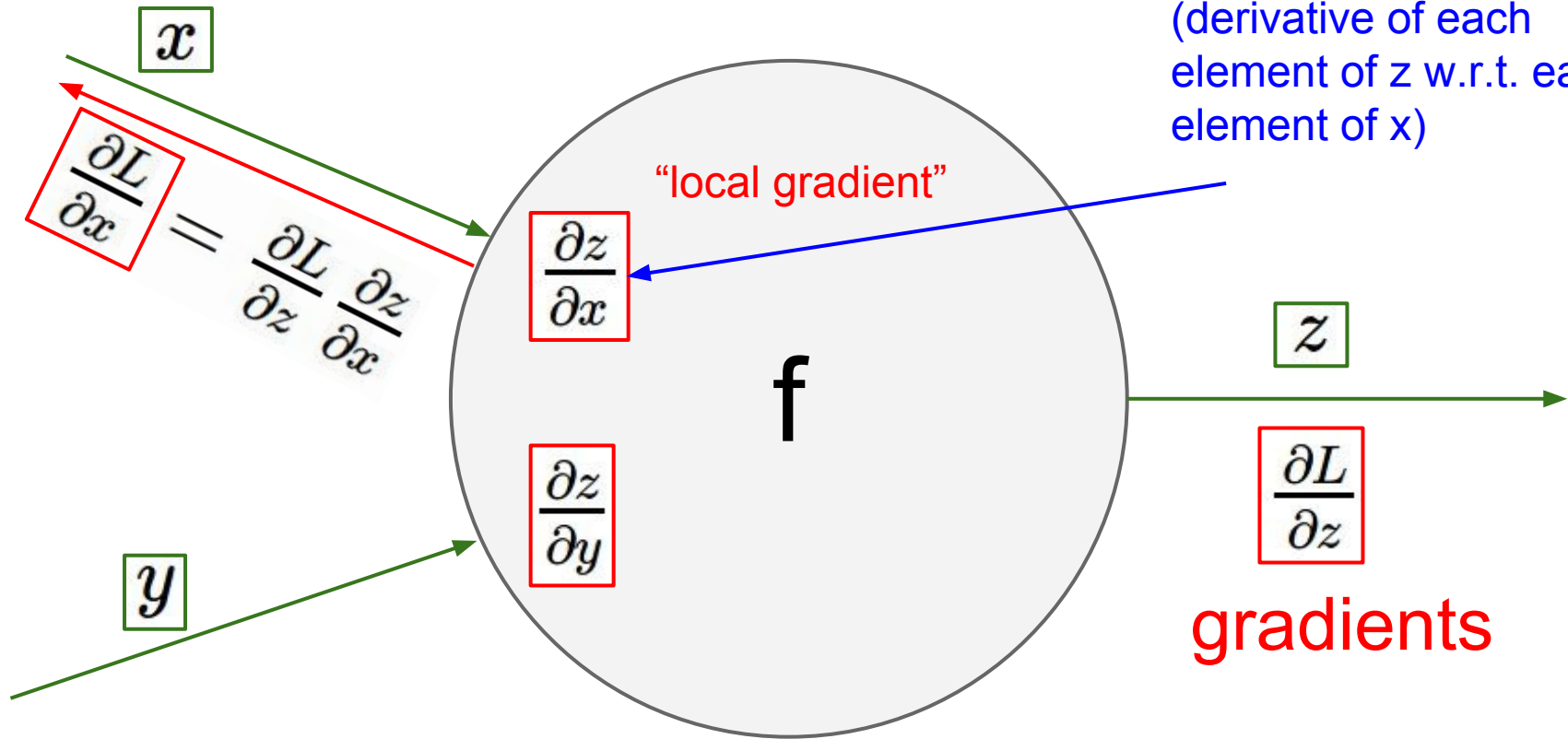
```
1 #include <cmath>
2 #include <vector>
3
4 #include "caffe/layers/sigmoid_layer.hpp"
5
6 namespace caffe {
7
8 template <typename Dtype>
9 inline Dtype sigmoid(Dtype x) {
10     return 1. / (1. + exp(-x));
11 }
12
13 template <typename Dtype>
14 void SigmoidLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
15     const vector<Blob<Dtype>*>& top) {
16     const Dtype* bottom_data = bottom[0]->cpu_data();
17     Dtype* top_data = top[0]->mutable_cpu_data();
18     const int count = bottom[0]->count();
19     for (int i = 0; i < count; ++i) {
20         top_data[i] = sigmoid(bottom_data[i]);
21     }
22 }
23
24 template <typename Dtype>
25 void SigmoidLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
26     const vector<bool>& propagate_down,
27     const vector<Blob<Dtype>*>& bottom) {
28     if (propagate_down[0]) {
29         const Dtype* top_data = top[0]->cpu_data();
30         const Dtype* top_diff = top[0]->cpu_diff();
31         Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
32         const int count = bottom[0]->count();
33         for (int i = 0; i < count; ++i) {
34             const Dtype sigmoid_x = top_data[i];
35             bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
36         }
37     }
38 }
39
40 #ifdef CPU_ONLY
41 STUB_GPU(SigmoidLayer);
42 #endif
43
44 INSTANTIATE_CLASS(SigmoidLayer);
45
46
47 } // namespace caffe
```

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

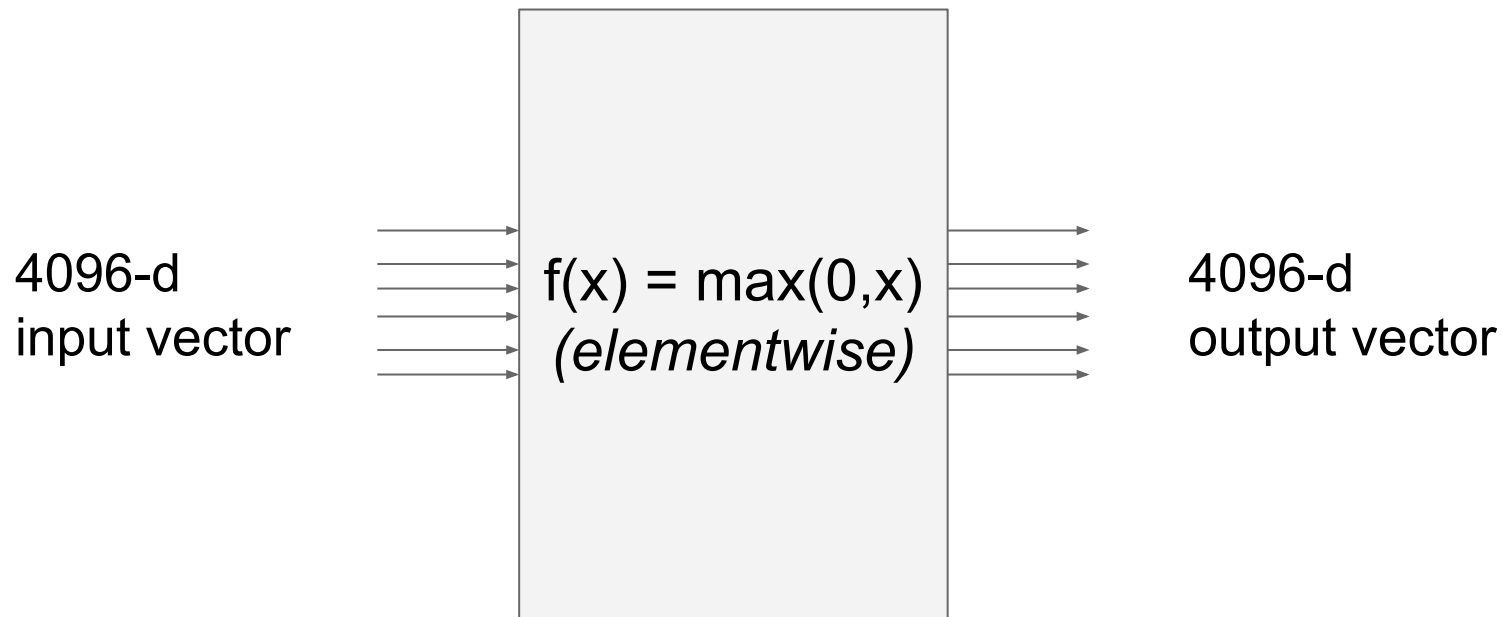
$$(1 - \sigma(x)) \sigma(x) \text{ *top\_diff (chain rule)}$$

# Gradients for vectorized code (x,y,z are now vectors)

This is now the **Jacobian matrix**  
(derivative of each element of z w.r.t. each element of x)



# Vectorized operations

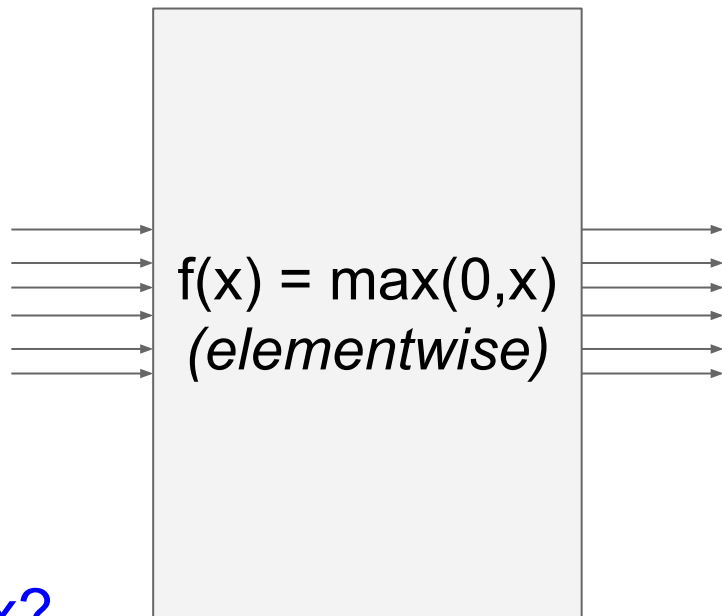


## Vectorized operations

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

Jacobian matrix

4096-d  
input vector



4096-d  
output vector

Q: what is the  
size of the  
Jacobian matrix?

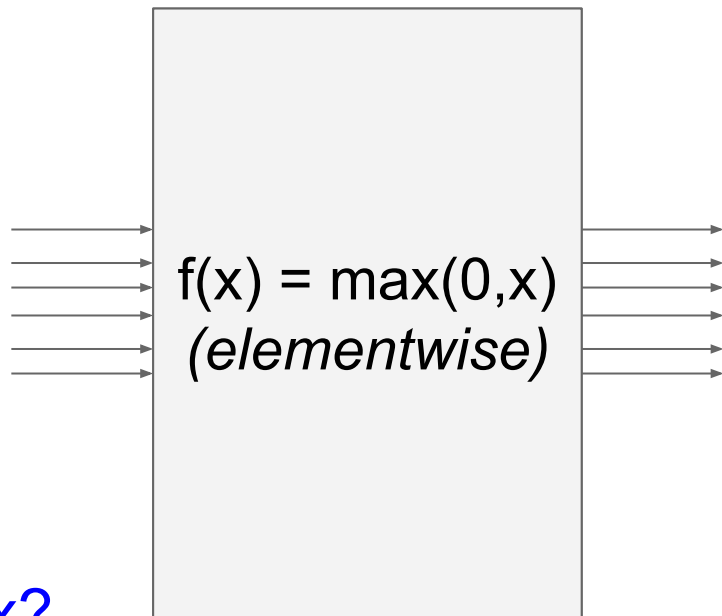


## Vectorized operations

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

Jacobian matrix

4096-d  
input vector



$f(x) = \max(0, x)$   
*(elementwise)*

4096-d  
output vector

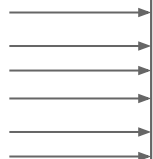
Q: what is the  
size of the  
Jacobian matrix?  
[4096 x 4096!]

Q2: what does it  
look like?

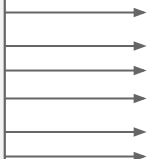
# Vectorized operations

in practice we process an entire minibatch (e.g. 100) of examples at one time:

100 4096-d  
input vectors



$f(x) = \max(0, x)$   
(*elementwise*)



100 4096-d  
output vectors

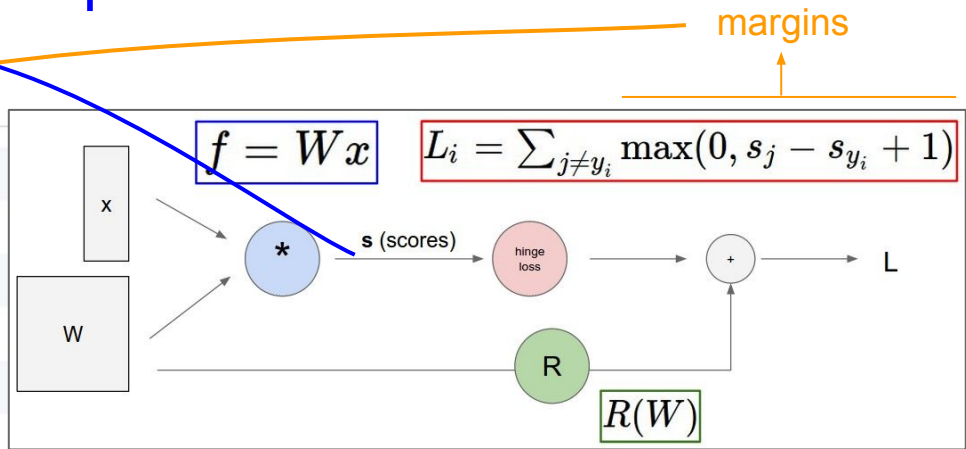
i.e. Jacobian would technically be a [409,600 x 409,600] matrix :)

# Assignment: Writing SVM/Softmax

## Stage your forward/backward computation!

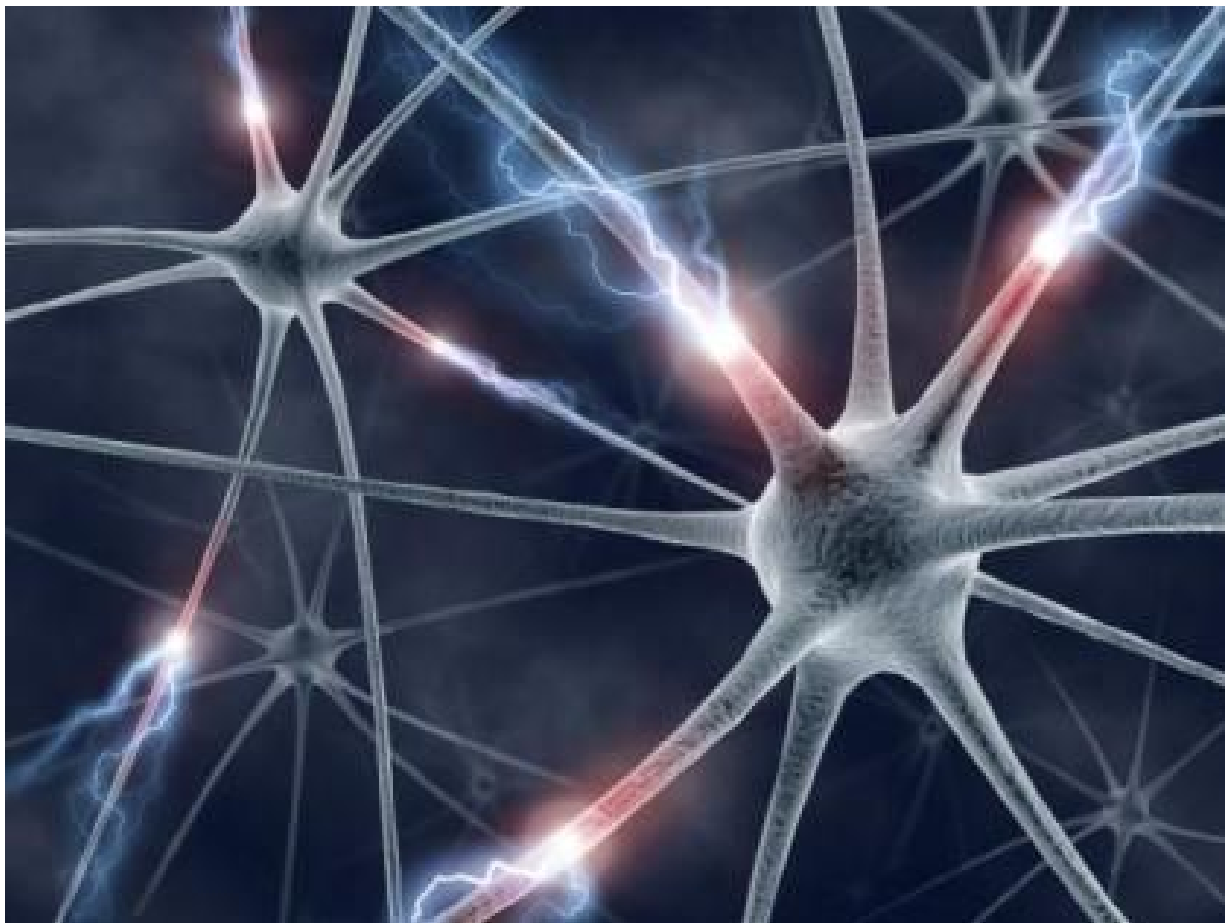
E.g. for the SVM:

```
# receive W (weights), X (data)
# forward pass (we have 8 lines)
scores = #...
margins = #...
data_loss = #...
reg_loss = #...
loss = data_loss + reg_loss
# backward pass (we have 5 lines)
dmargins = # ... (optionally, we go direct to dscores)
dscores = #...
dW = #...
```



# Summary so far

- neural nets will be very large: no hope of writing down gradient formula by hand for all parameters
- **backpropagation** = recursive application of the chain rule along a computational graph to compute the gradients of all inputs/parameters/intermediates
- implementations maintain a graph structure, where the nodes implement the **forward()** / **backward()** API.
- **forward**: compute result of an operation and save any intermediates needed for gradient computation in memory
- **backward**: apply the chain rule to compute the gradient of the loss function with respect to the inputs.



# Neural Network: without the brain stuff

(**Before**) Linear score function:  $f = Wx$

# Neural Network: without the brain stuff

(**Before**) Linear score function:

$$f = Wx$$

(**Now**) 2-layer Neural Network

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

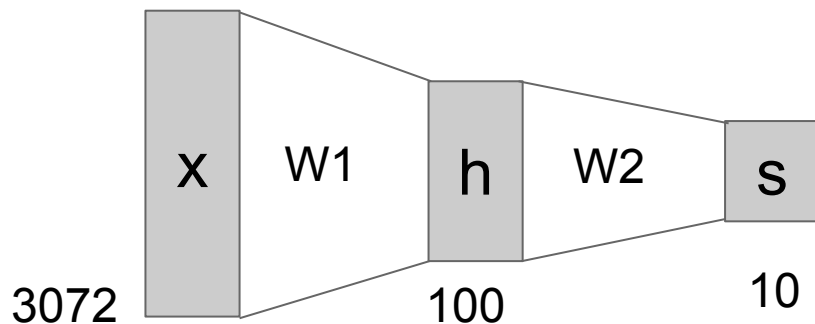
# Neural Network: without the brain stuff

(**Before**) Linear score function:

$$f = Wx$$

(**Now**) 2-layer Neural Network

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





# Neural Network: without the brain stuff

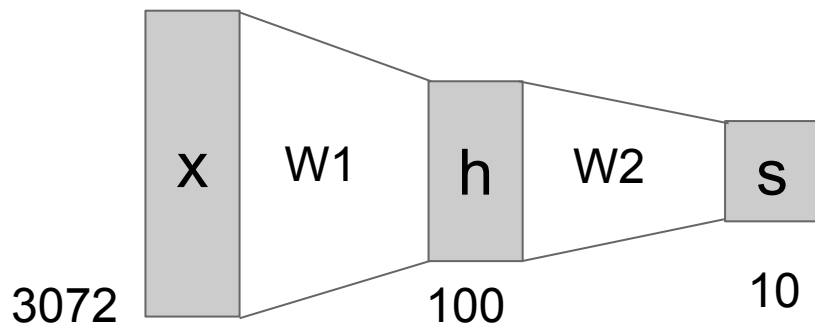


(Before) Linear score function:

$$f = Wx$$

(Now) 2-layer Neural Network

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



# Neural Network: without the brain stuff

(**Before**) Linear score function:  $f = Wx$

(**Now**) 2-layer Neural Network  
or 3-layer Neural Network  $f = W_2 \max(0, W_1 x)$

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Full implementation of training a 2-layer Neural Network needs ~11 lines:

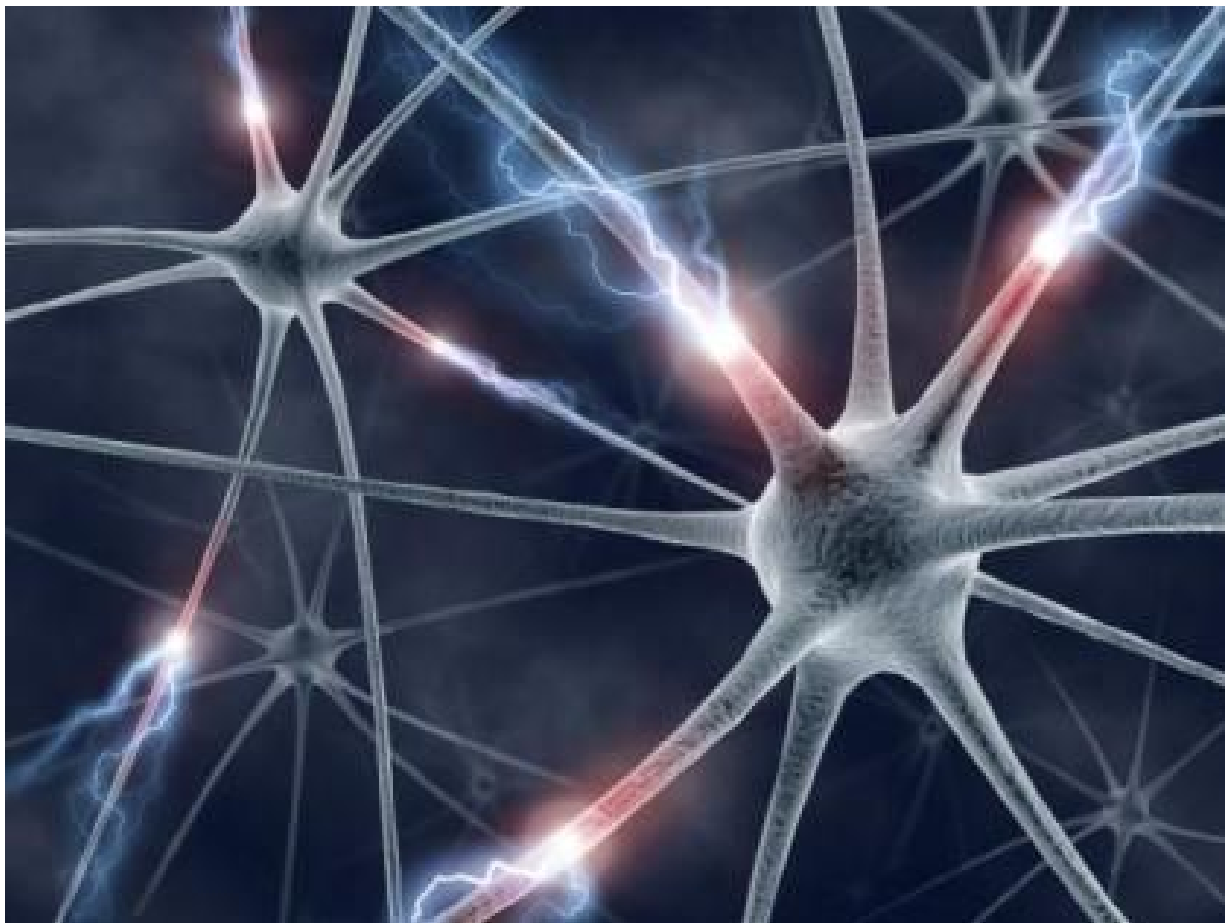
```
01. X = np.array([ [0,0,1], [0,1,1], [1,0,1], [1,1,1] ])
02. y = np.array([[0,1,1,0]]).T
03. syn0 = 2*np.random.random((3,4)) - 1
04. syn1 = 2*np.random.random((4,1)) - 1
05. for j in xrange(60000):
06.     l1 = 1/(1+np.exp(-(np.dot(X, syn0))))
07.     l2 = 1/(1+np.exp(-(np.dot(l1, syn1))))
08.     l2_delta = (y - l2)*(l2*(1-l2))
09.     l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
10.     syn1 += l1.T.dot(l2_delta)
11.     syn0 += X.T.dot(l1_delta)
```

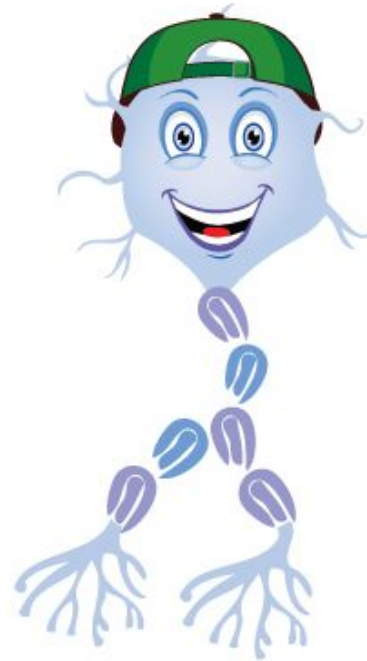
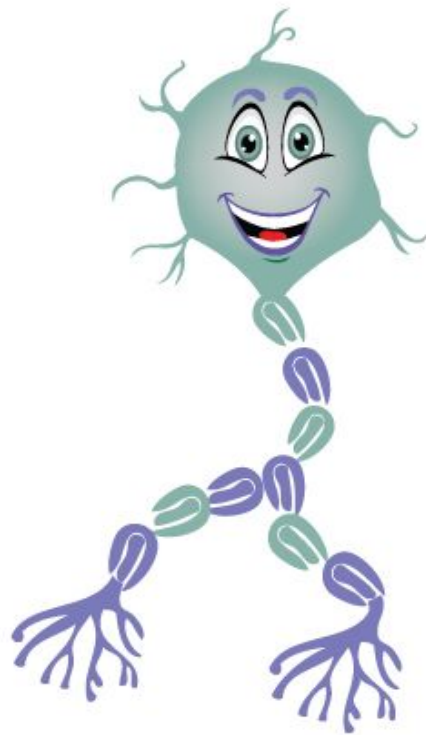
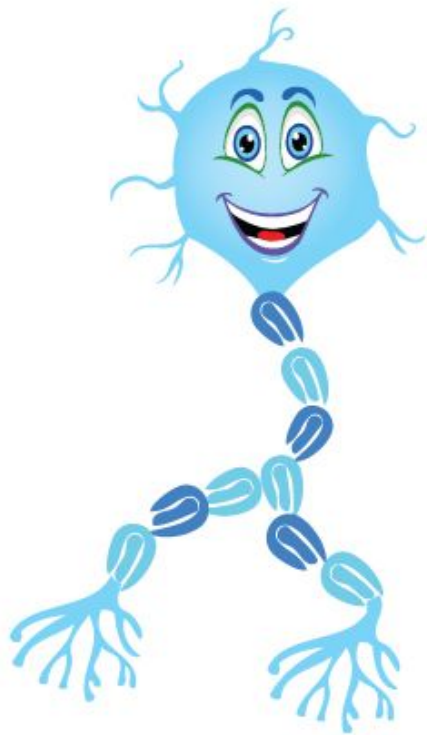
from @iamtrask, <http://iamtrask.github.io/2015/07/12/basic-python-network/>

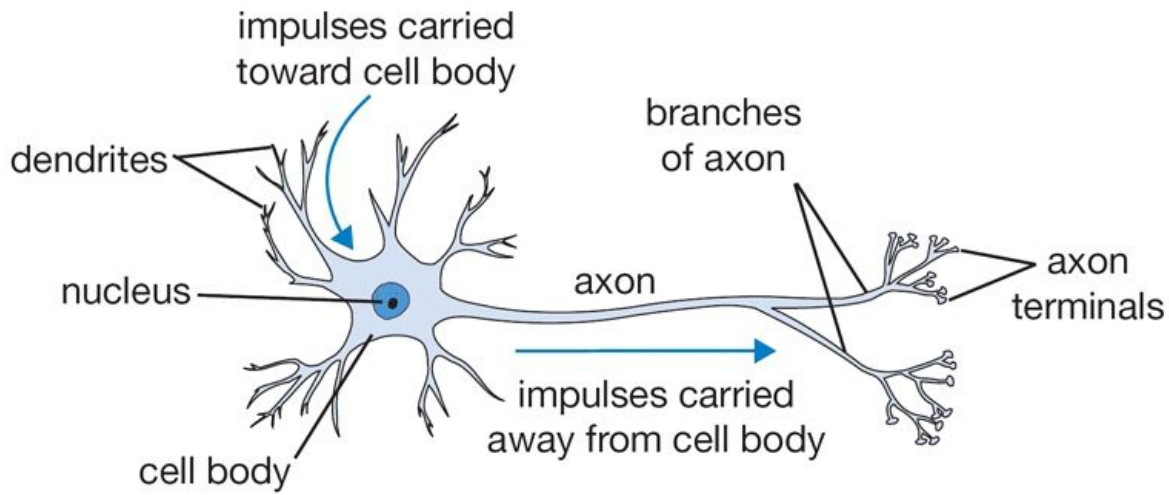
# Assignment: Writing 2layer Net

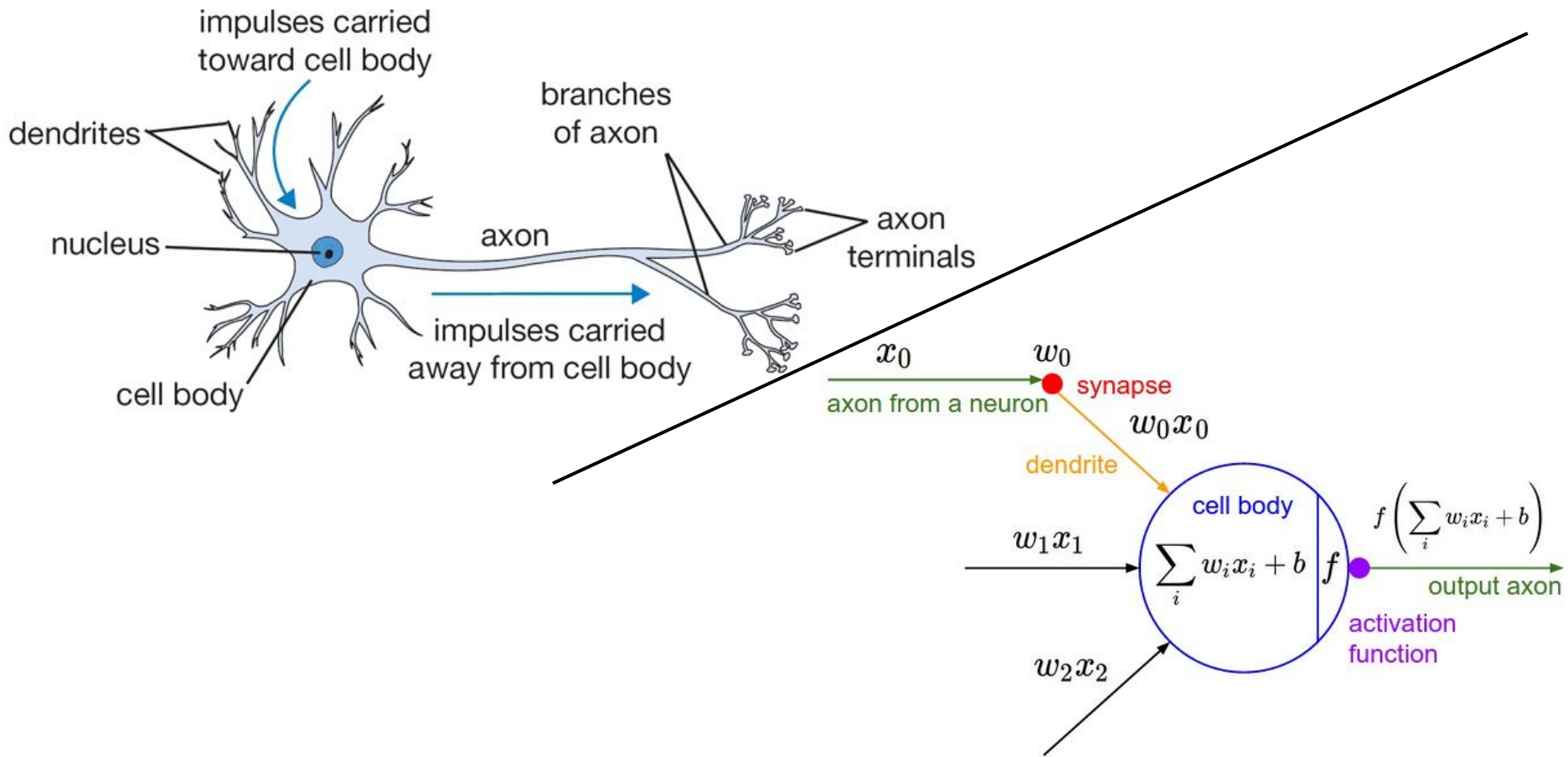
Stage your forward/backward computation!

```
# receive W1,W2,b1,b2 (weights/biases), X (data)
# forward pass:
h1 = #... function of X,W1,b1
scores = #... function of h1,W2,b2
loss = #... (several lines of code to evaluate Softmax loss)
# backward pass:
dscores = #...
dh1,dW2,db2 = #...
dW1,db1 = #...
```

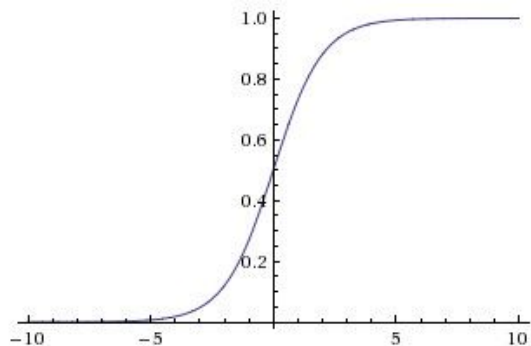
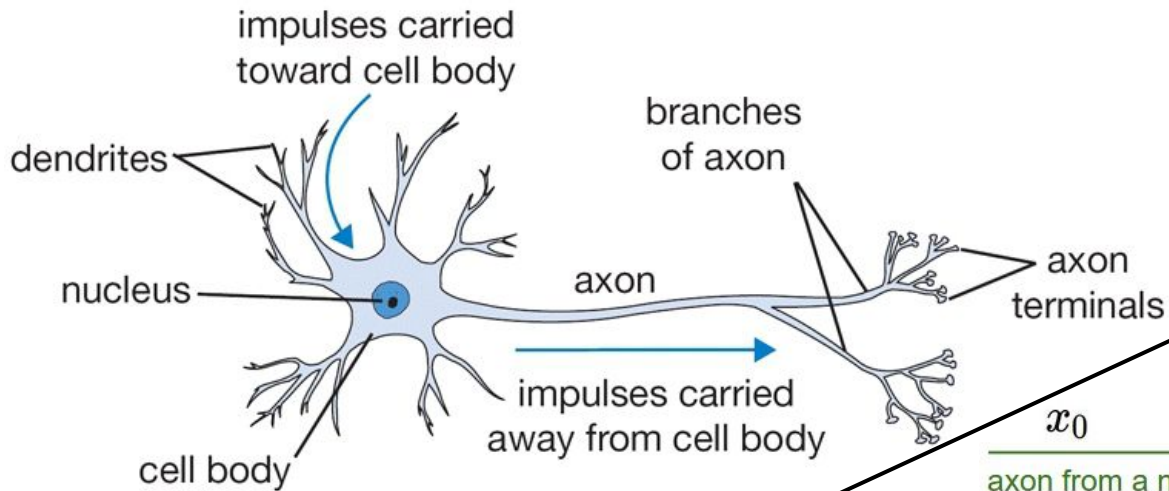






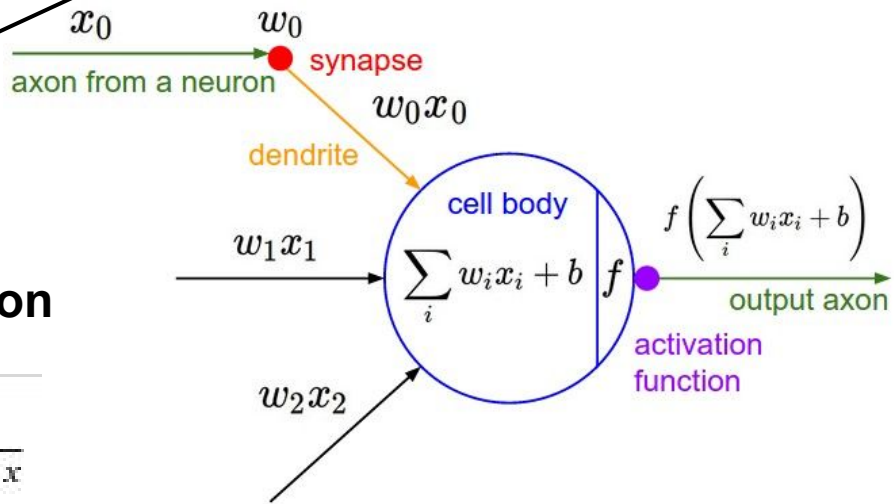


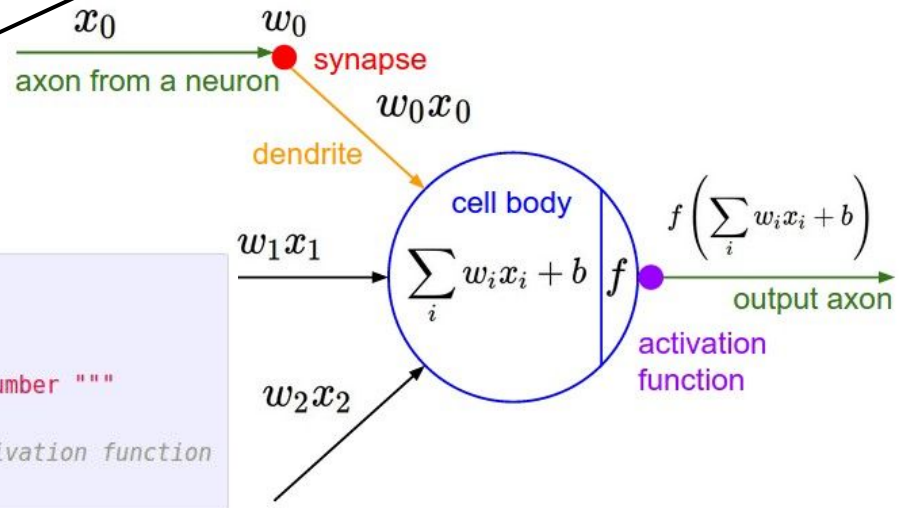
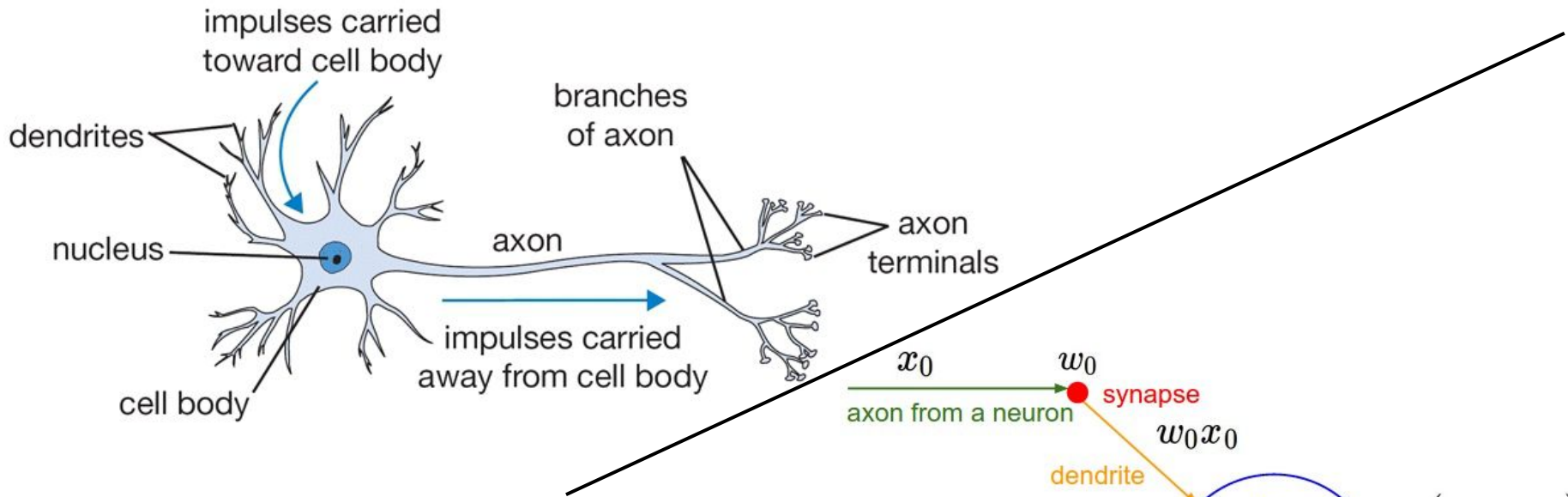




**sigmoid activation function**

$$\frac{1}{1 + e^{-x}}$$



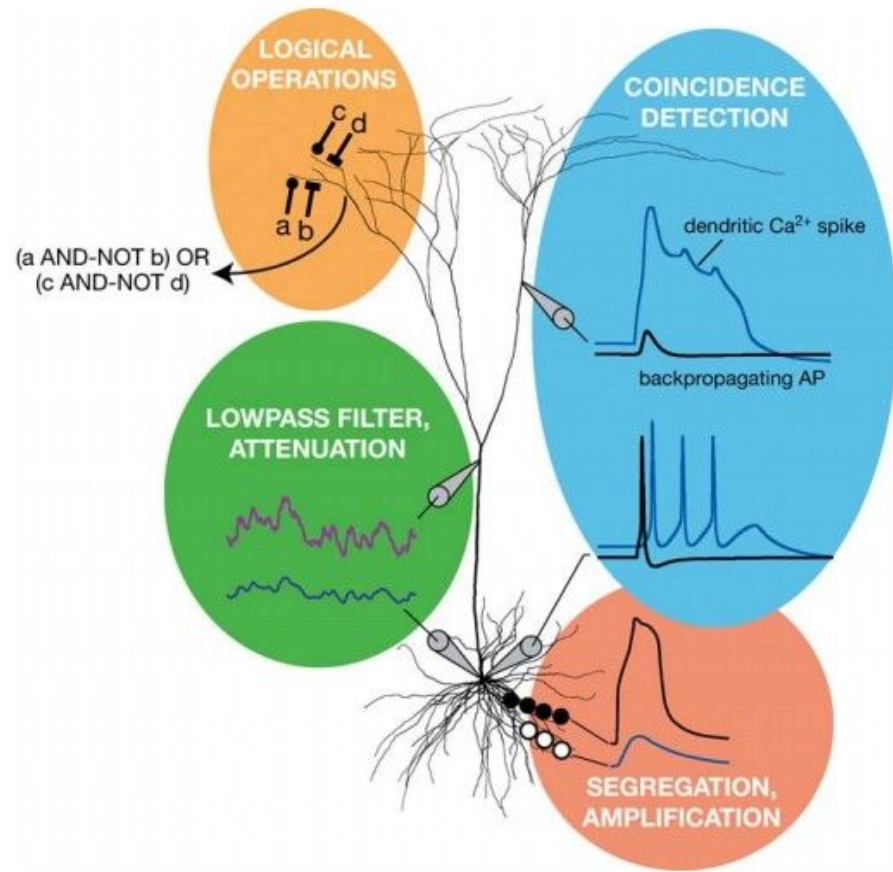


```
class Neuron:
    # ...
    def neuron_tick(inputs):
        """ assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
        return firing_rate
```

Be very careful with your Brain analogies:

### Biological Neurons:

- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

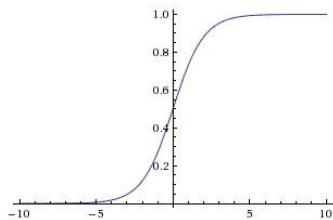


*[Dendritic Computation. London and Hausser]*

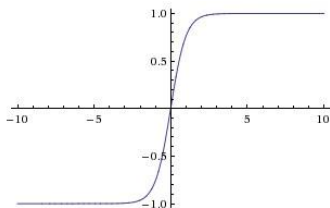
# Activation Functions

## Sigmoid

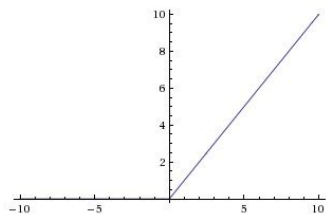
$$\sigma(x) = 1/(1 + e^{-x})$$



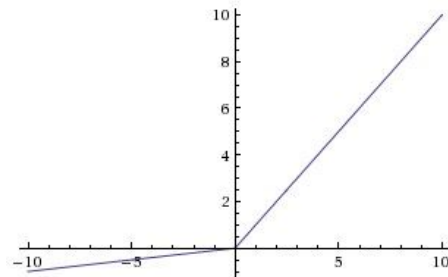
## tanh tanh(x)



## ReLU max(0,x)



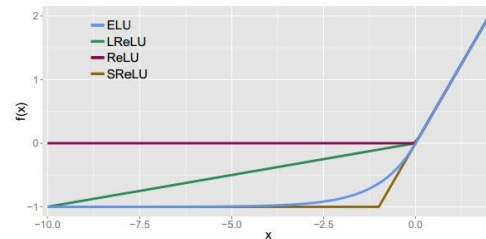
## Leaky ReLU max(0.1x, x)



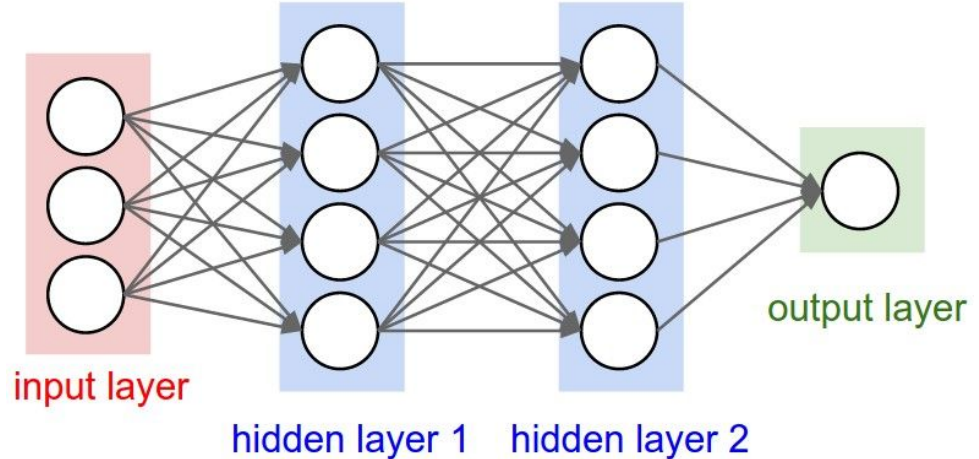
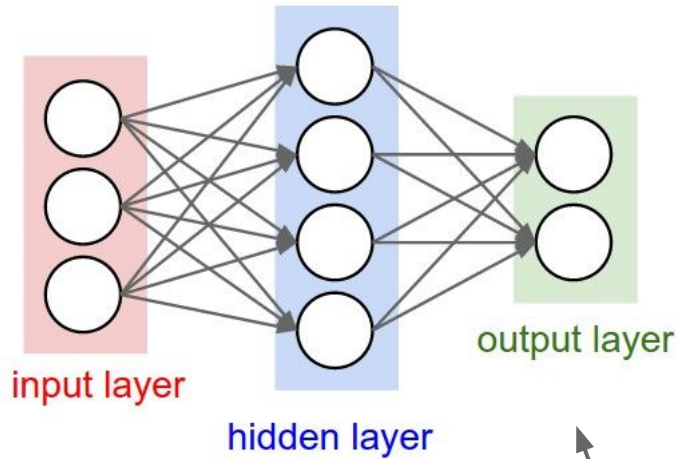
## Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

## ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$



# Neural Networks: Architectures



"2-layer Neural Net", or  
"1-hidden-layer Neural Net"

"3-layer Neural Net", or  
"2-hidden-layer Neural Net"

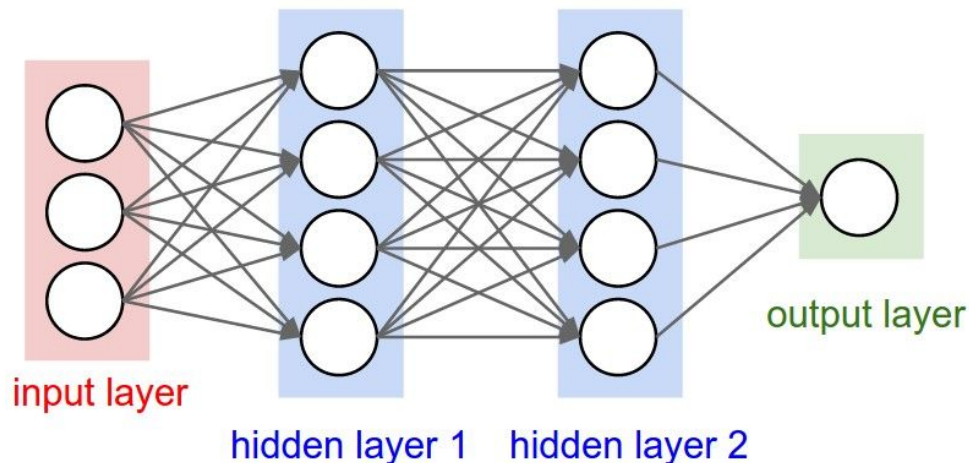
**"Fully-connected" layers**

# Example Feed-forward computation of a Neural Network

```
class Neuron:
    # ...
    def neuron_tick(inputs):
        """ assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
        return firing_rate
```

We can efficiently evaluate an entire layer of neurons.

# Example Feed-forward computation of a Neural Network



```
# forward-pass of a 3-layer neural network:
```

```
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
```

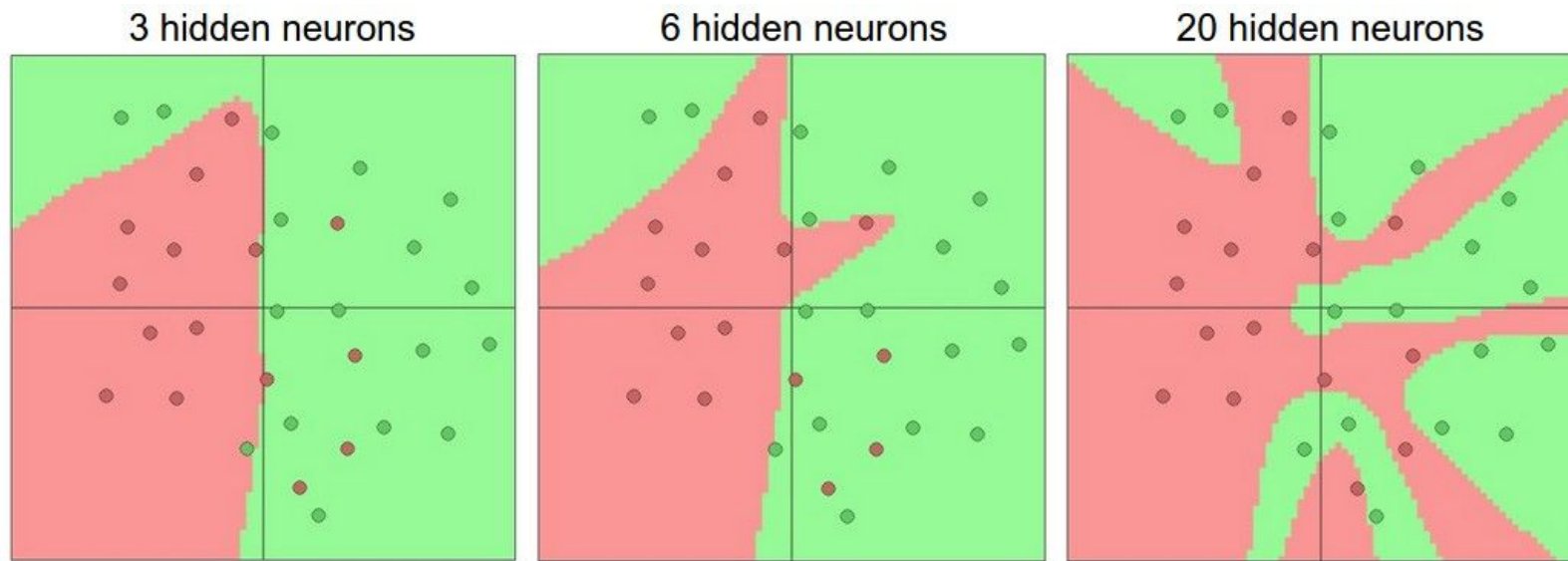
```
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
```

```
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
```

```
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
```

```
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

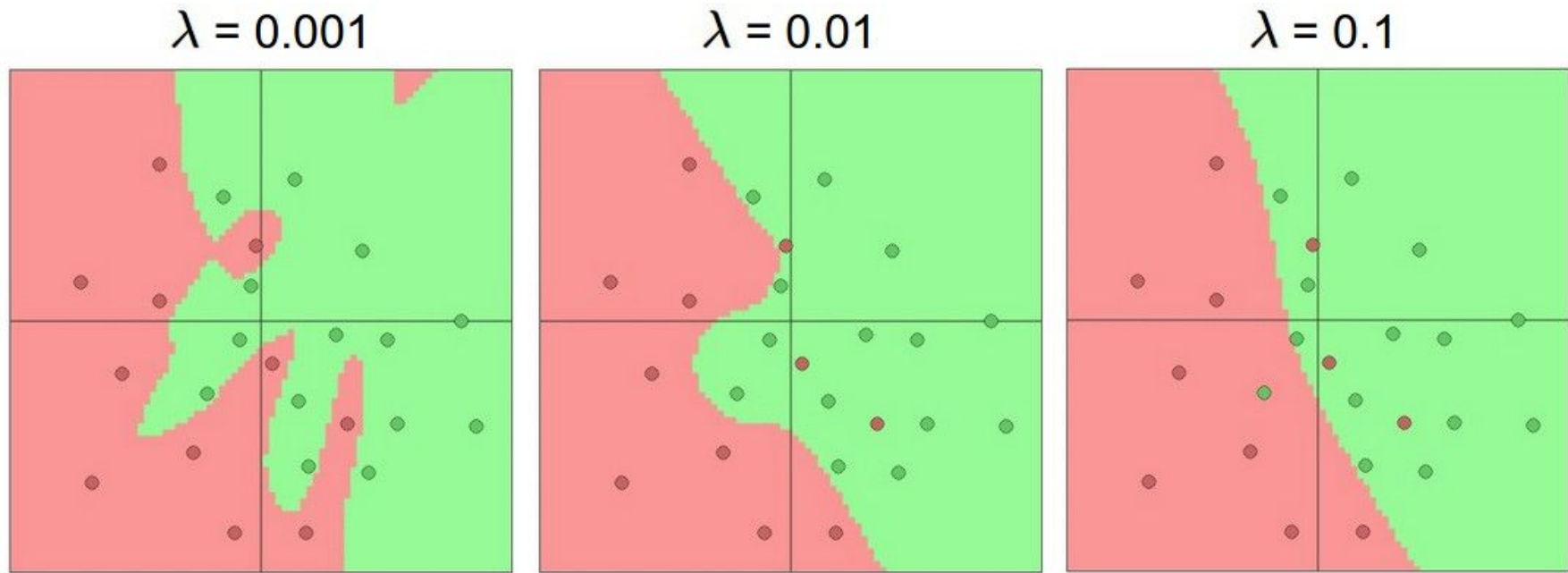
# Setting the number of layers and their sizes



more neurons = more capacity



Do not use size of neural network as a regularizer. Use stronger regularization instead:



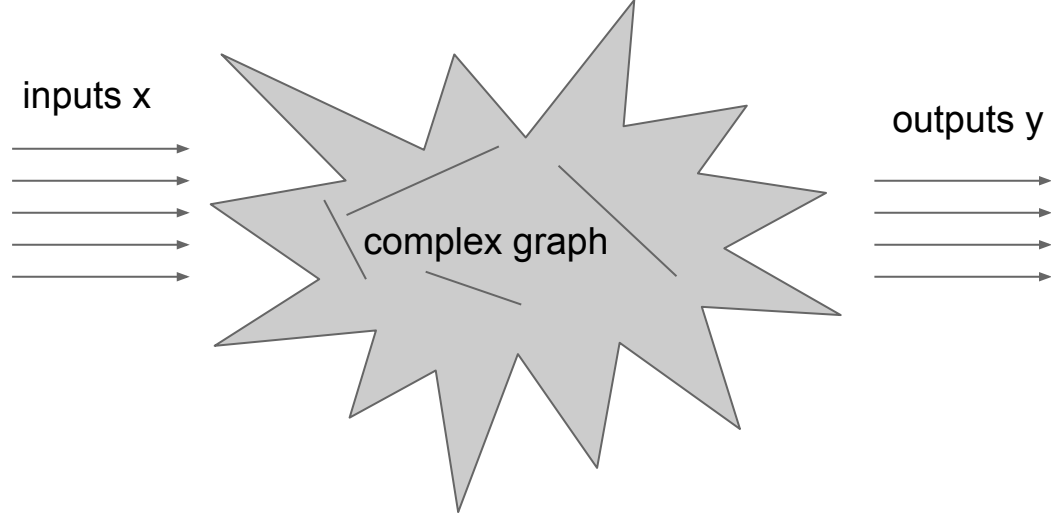
(you can play with this demo over at ConvNetJS: <http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>)

# Summary

- we arrange neurons into fully-connected layers
- the abstraction of a **layer** has the nice property that it allows us to use efficient vectorized code (e.g. matrix multiplies)
- neural networks are not really *neural*
- neural networks: bigger = better (but might have to regularize more strongly)

## **Next Lecture:**

More than you ever wanted to know about Neural Networks and how to train them.



← reverse-mode differentiation (if you want effect of many things on one thing)

$$\frac{\partial y}{\partial x} \text{ for many different } x$$

→ forward-mode differentiation (if you want effect of one thing on many things)

$$\frac{\partial y}{\partial x} \text{ for many different } y$$