Lecture 4:
Backpropagation
and
Neural Networks (part 1)

Tuesday January 31, 2017

Announcements!

- If you are adversely affected by immigration ban, please talk to me about accommodations
- Send in paper choices by **tonight**
- Should be able to run Jupyter server on Tufts was and network machines now
    - (deep-venv)> pip install --upgrade jupyter
- hw1 deadline in two days — Thurs Feb 2: Don't forget to read the course notes.
- Redo calculation of dL/dW for hinge loss

Python/Numpy of the Day

- `y_pred = scores.argmax(axis=1)`
- `inds = np.random.choice(X.shape[0],batch_size)`
  - randomly select N numbers in a range,
  - useful for subsampling
- `[:,np.newaxis]`
  - reshapes matrices of size (N,) to size (N,1)

Where we are...

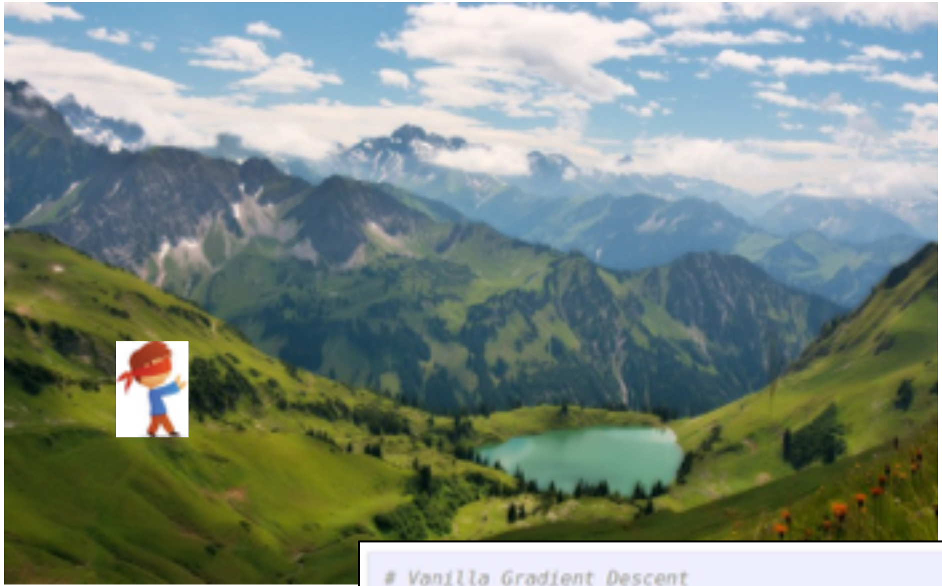
$$s = f(x; W) = Wx$$ scores function

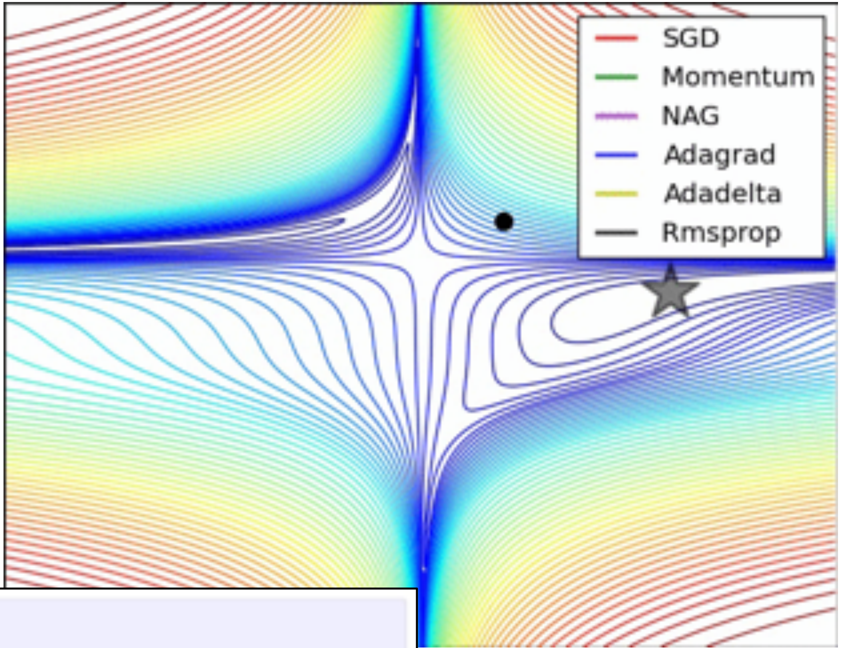
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$ SVM loss

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

want $\boxed{\nabla_W L}$

Optimization






```python
# 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

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

Numerical gradient: slow :(, approximate :(, easy to write :)
Analytic gradient: fast :), exact :), error-prone :(

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

Hinge Loss Gradient wrt Weights W

margin size, usually 1.0

$$L_i = \sum_{j \neq y_i} \left[\max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)\right]$$

- We want the Jacobian Matrix of all gradients
  - partial derivatives of all output dimensions by all input dimensions

$$\nabla w L = \begin{bmatrix} \nabla w_1 L_1 & \dots & \dots & \nabla w_1 L_N \\ \vdots & \nabla w_j L_i & \ddots & \vdots \\ \nabla w_k L_1 & \dots & \dots & \nabla w_k L_N \end{bmatrix}$$

For all rows of *dW* where the row corresponds to the GT value for that training instance, i.e. $j = y_i$

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0)\right) x_i$$

For all rows of *dW* where $j \neq y_i$

$$\nabla_{w_j} L_i = \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) x_i$$

Softmax Loss Gradient wrt Score S

* note change of subscripts from last slide

$$a_j = w_j^T x_j$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^N e^{a_k}} \qquad \forall j \in 1..N$$

$$\frac{\partial S_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}}{\partial a_j}$$

$$\nabla a_j S_i, \text{ when } i = j$$

$$\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}}{\partial a_j} = \frac{e^{a_i}\Sigma - e^{a_j}e^{a_i}}{\Sigma^2}$$

$$= \frac{e^{a_i}}{\Sigma} \frac{\Sigma - e^{a_j}}{\Sigma}$$

$$= S_i(1 - S_j)$$

$$\nabla a_j S_i, \text{ when } i \neq j$$

$$\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}}{\partial a_j} = \frac{0 - e^{a_j}e^{a_i}}{\Sigma^2}$$

$$= -\frac{e^{a_j}}{\Sigma} \frac{e^{a_i}}{\Sigma}$$

$$= -S_j S_i$$

Skipping some steps for space,
please see original notes.

$$\nabla a_j S_i = S_i(\mathbb{1}(i = j) - S_j)$$

Softmax Loss Gradient wrt Score S

$$a_j = w_j^T x_j$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \qquad \forall j \in 1..N$$

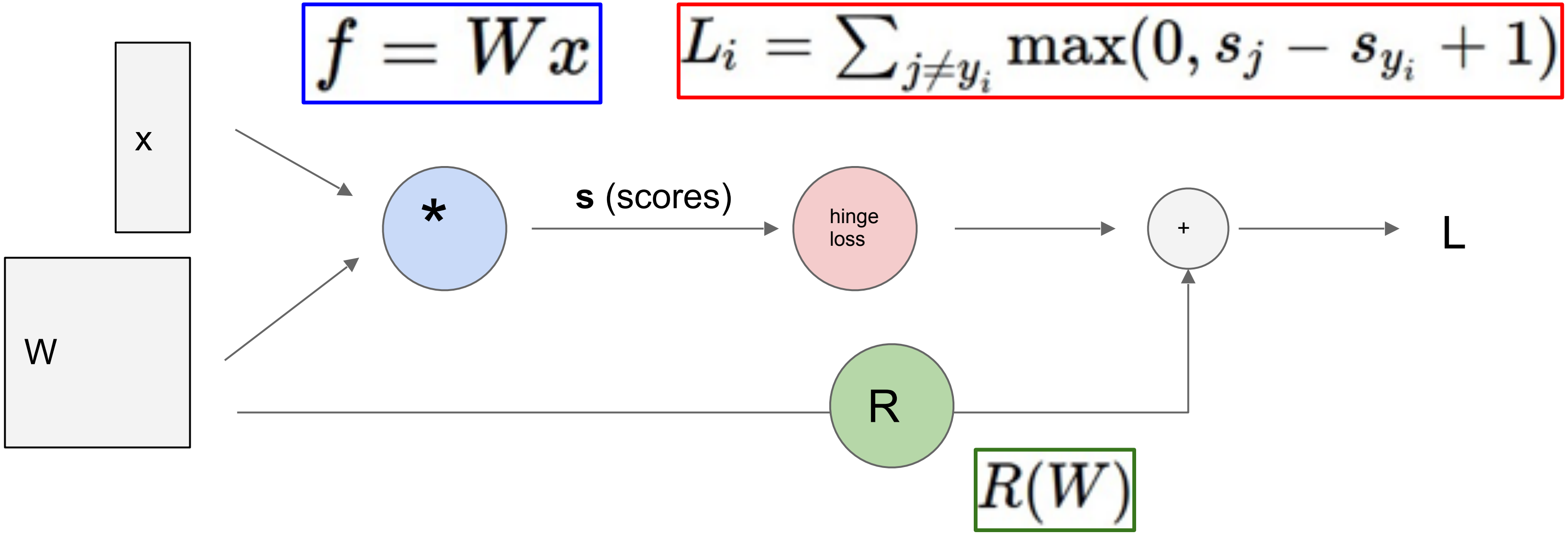
$$\nabla a_j S_i = S_i(\mathbb{1}(i = j) - S_j)$$

$$\nabla S_i L = \frac{\partial}{\partial S_i} - log(S_i) = S_j - \mathbb{1}(i = j)$$

$$\nabla W_j L = \frac{\partial L}{\partial S_i} * \frac{\partial S_i}{\partial W_j} = (S_j - \mathbb{1}(i = j))x_i$$

Skipping some steps for space,
please see original notes.

Computational Graph

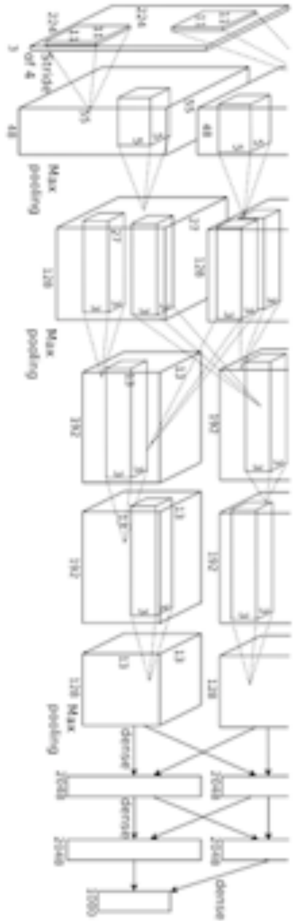


Convolutional Network (AlexNet)

input image

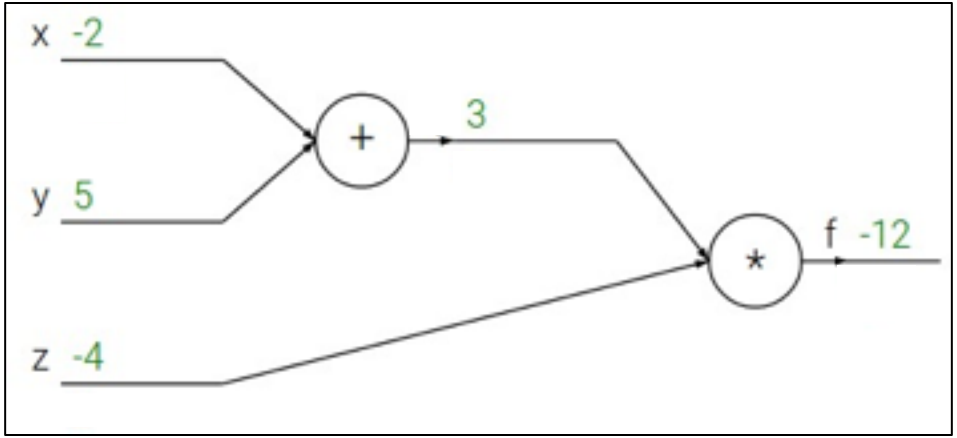
weights

loss



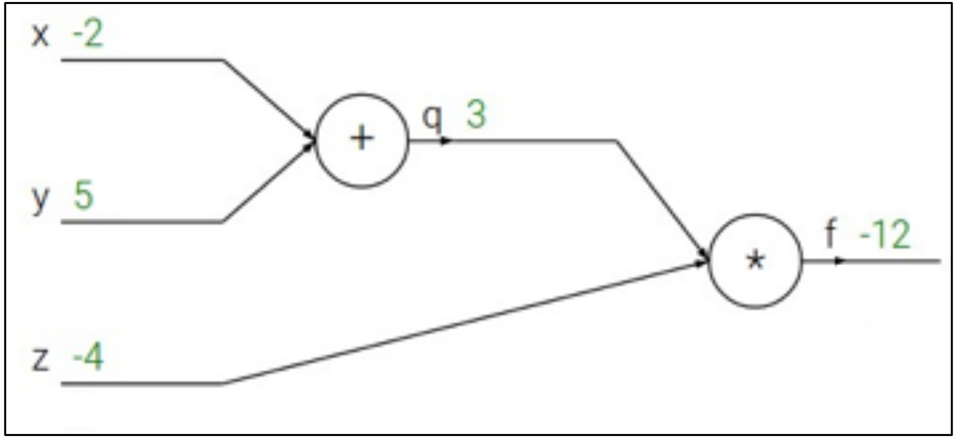
$$f(x,y,z) = (x+y)z$$

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



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

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



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

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

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

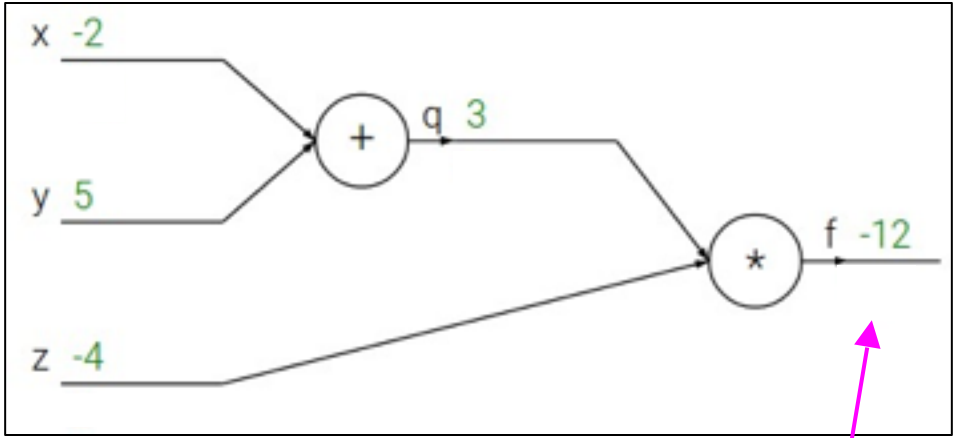
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

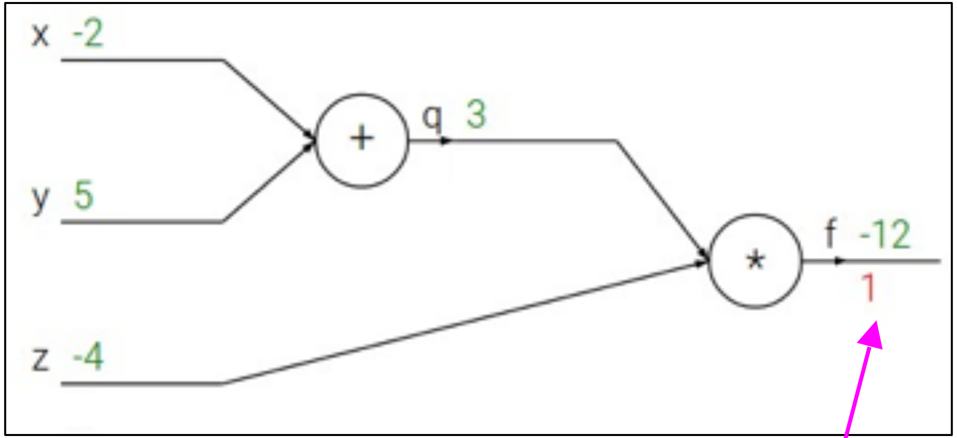
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

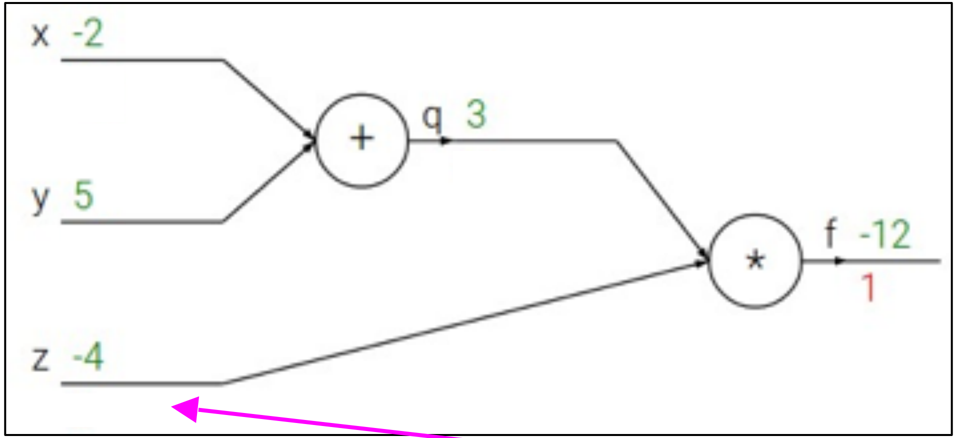
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

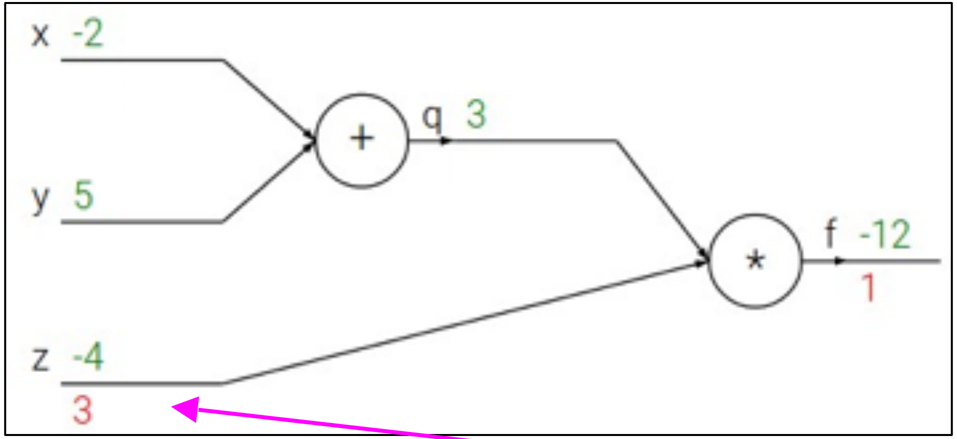
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

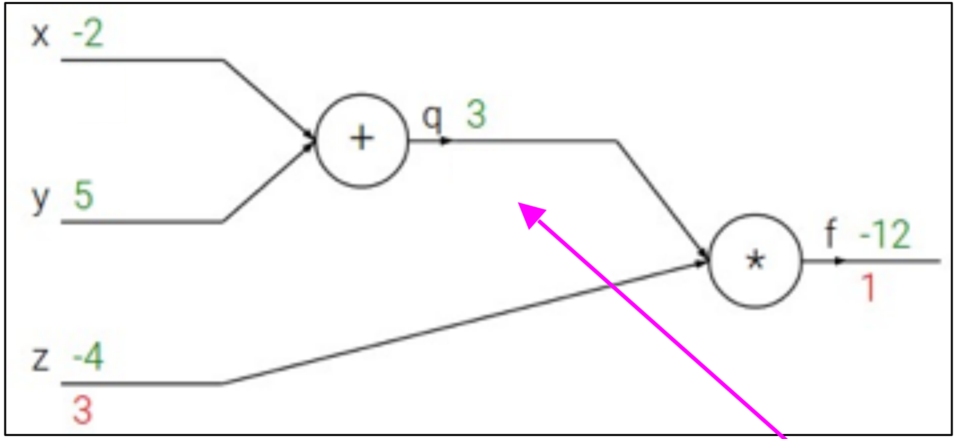
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

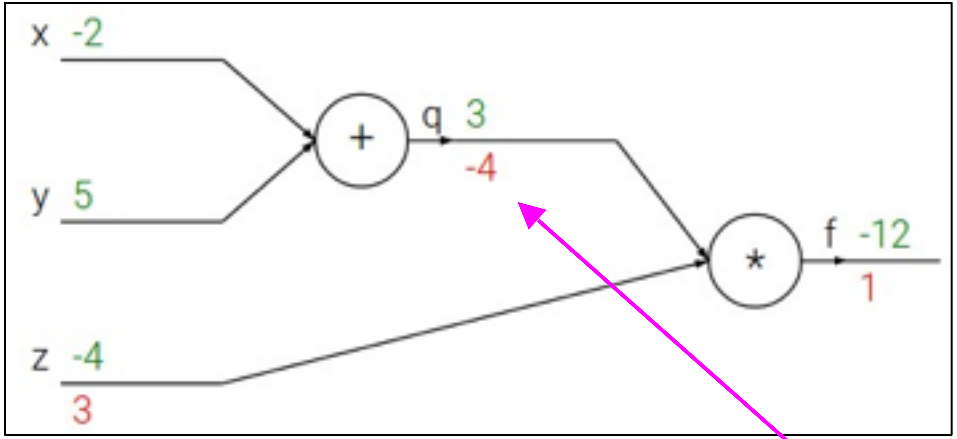
$$f(x, y, z) = (x + y)z$$

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

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

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

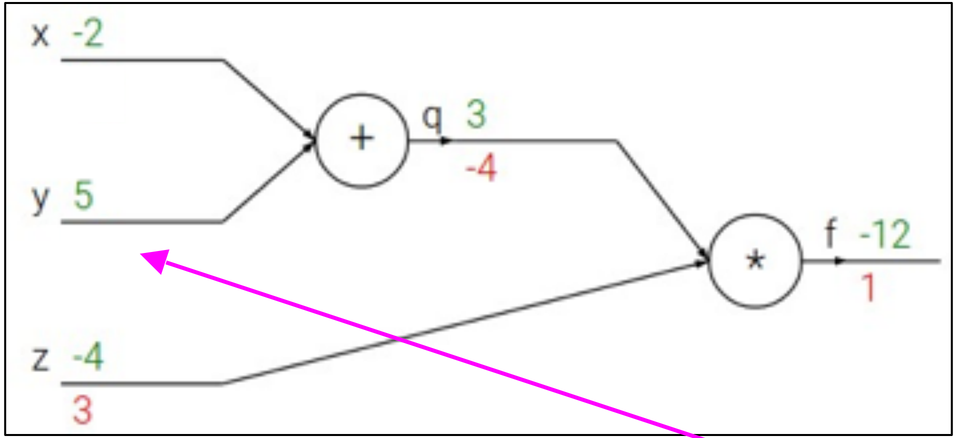
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



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

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

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



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

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

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

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

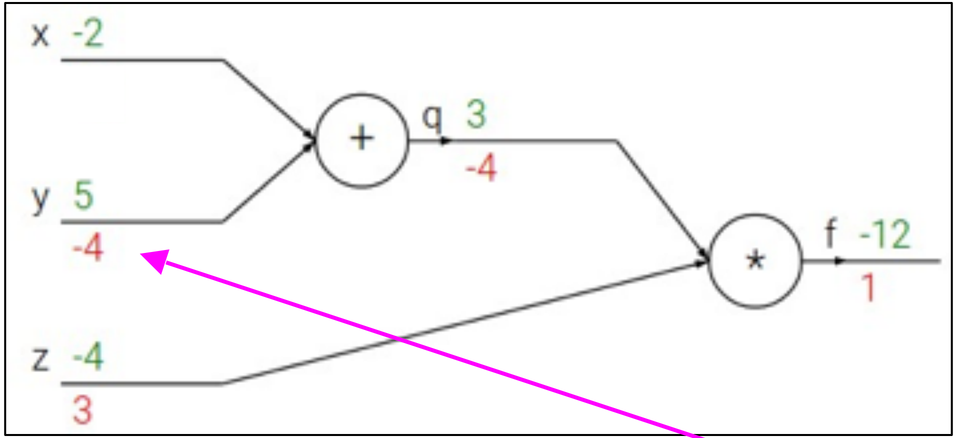
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

Chain rule:

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

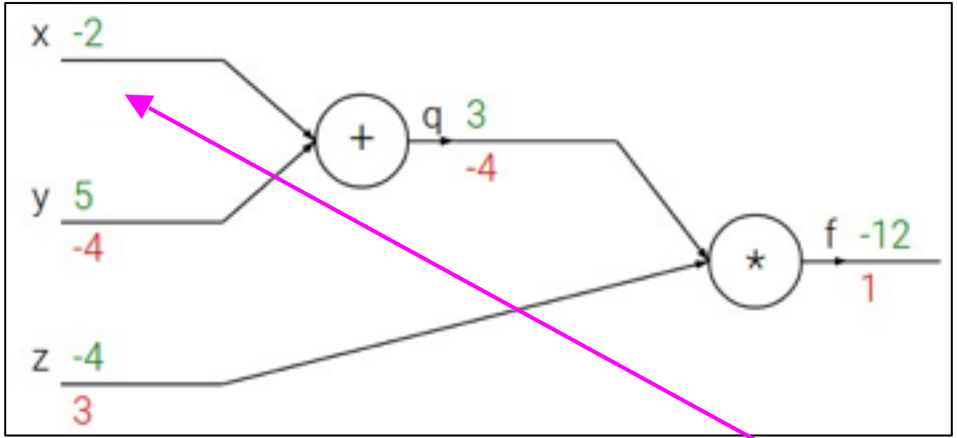
$$f(x,y,z) = (x+y)z$$

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

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

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

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



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

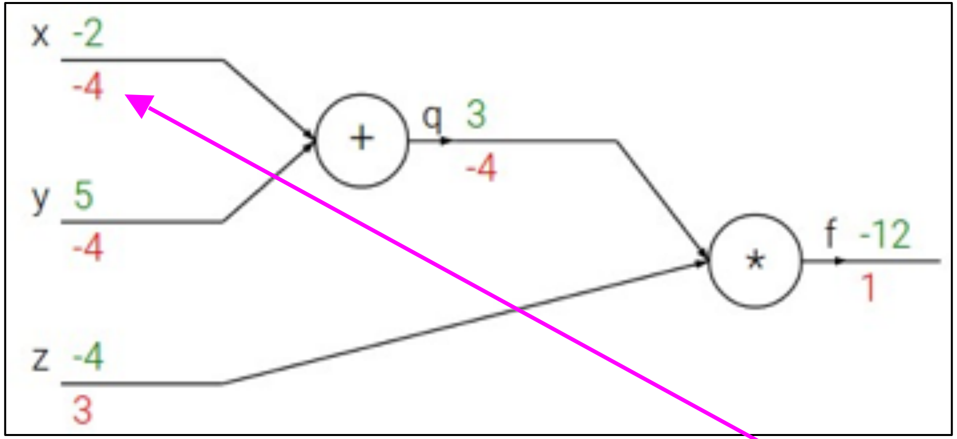
$$f(x, y, z) = (x + y)z$$

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

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

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

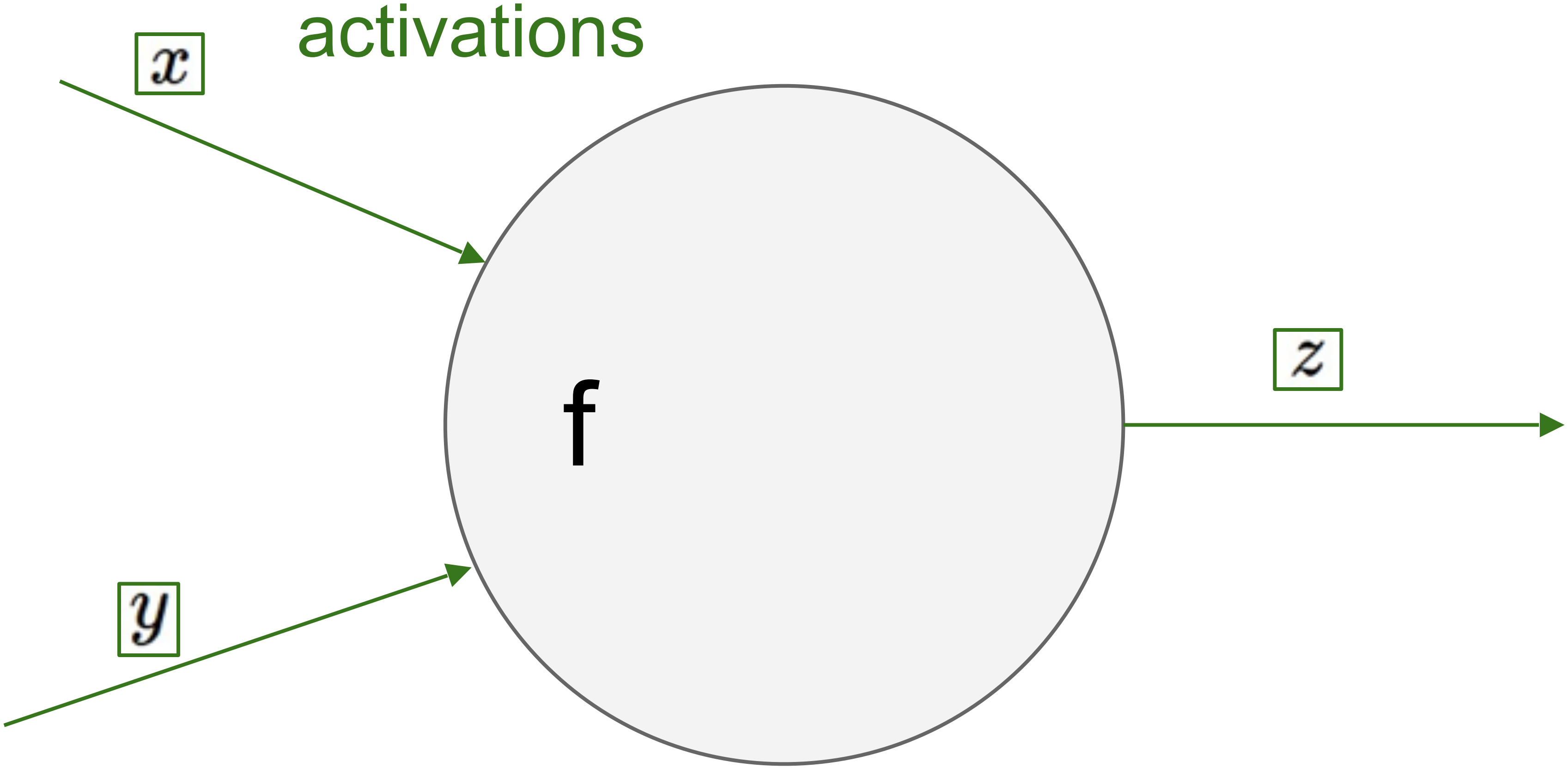
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



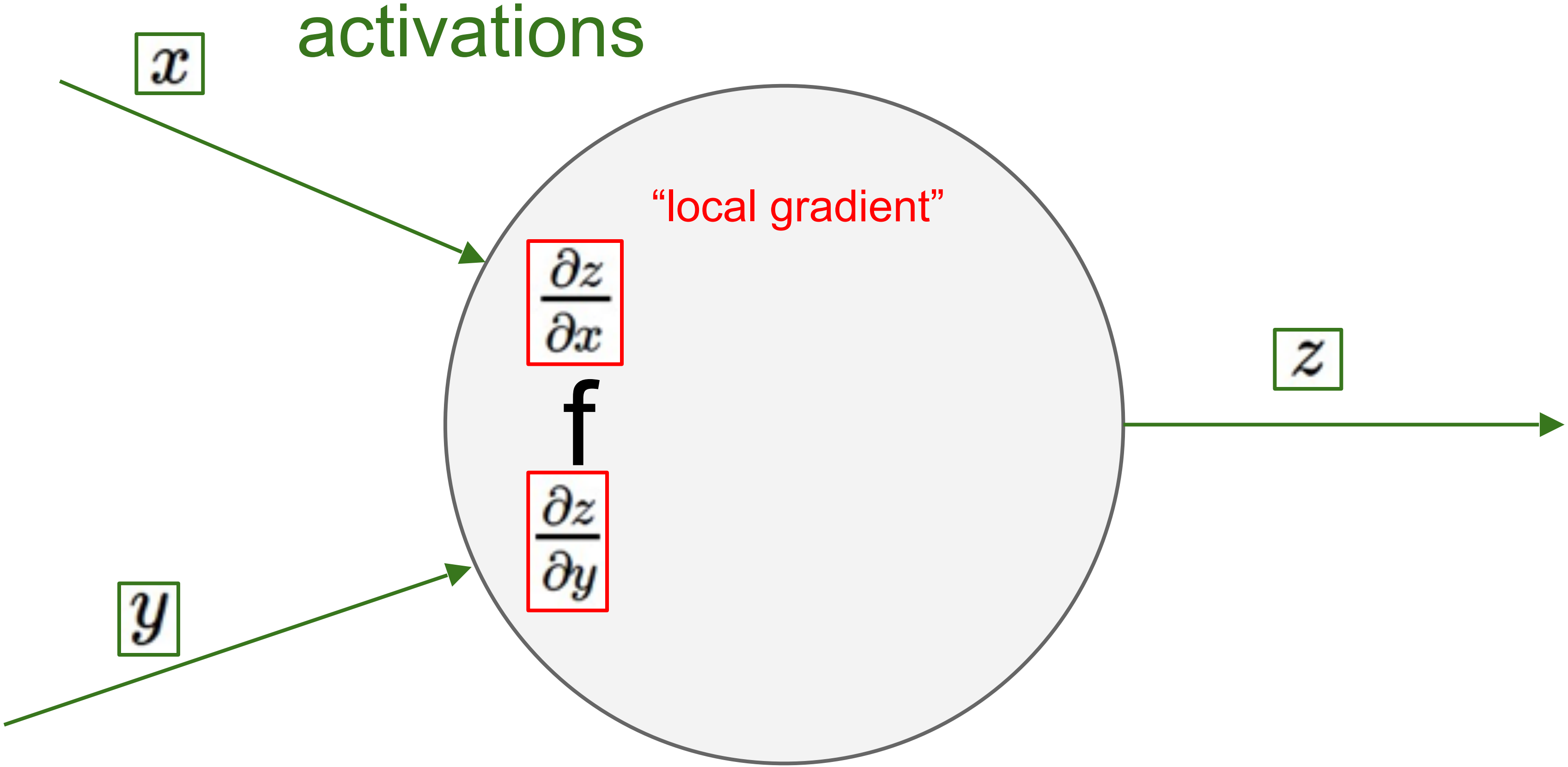
$\boxed{\frac{\partial f}{\partial x}}$

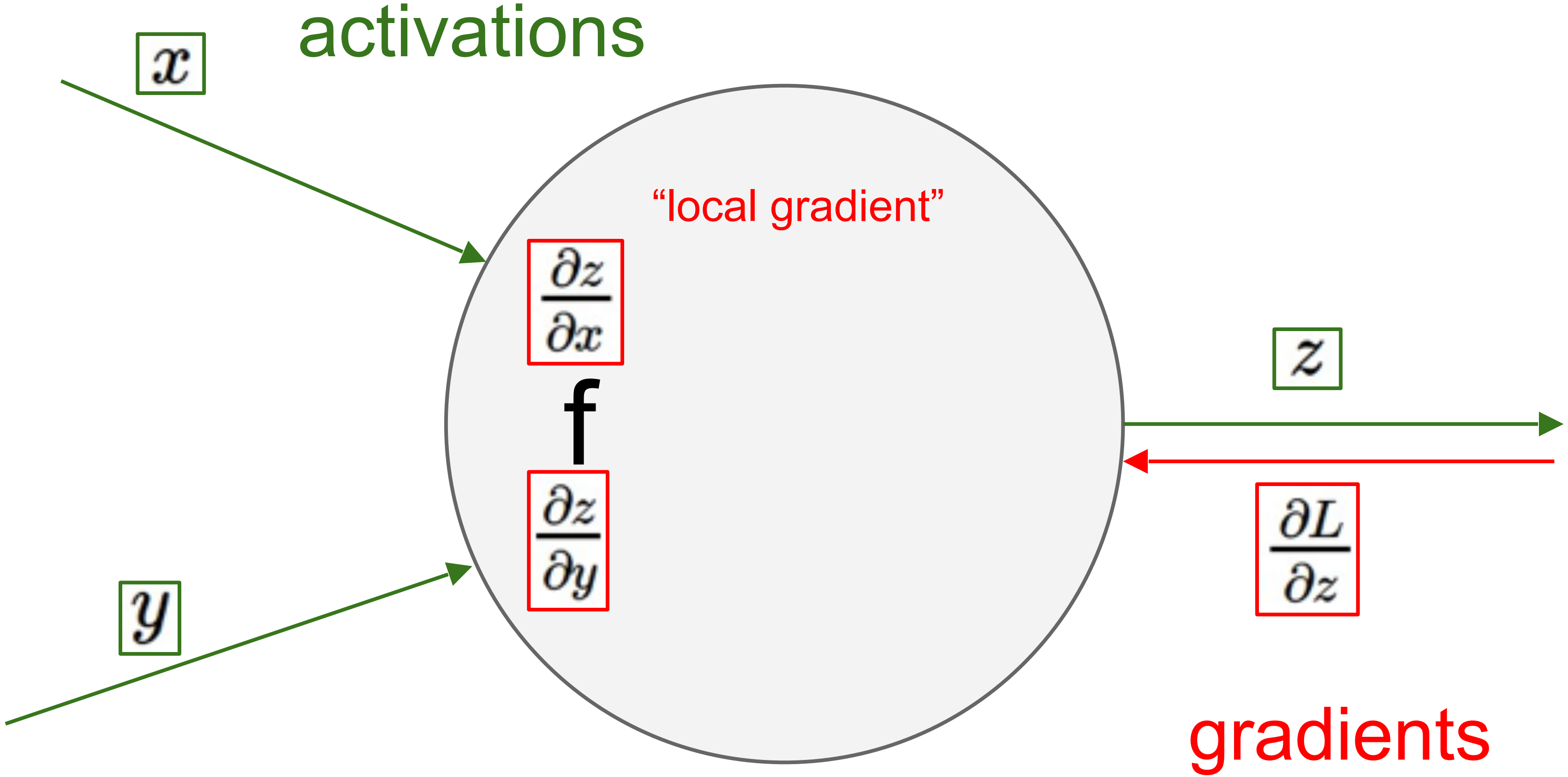
Chain rule:

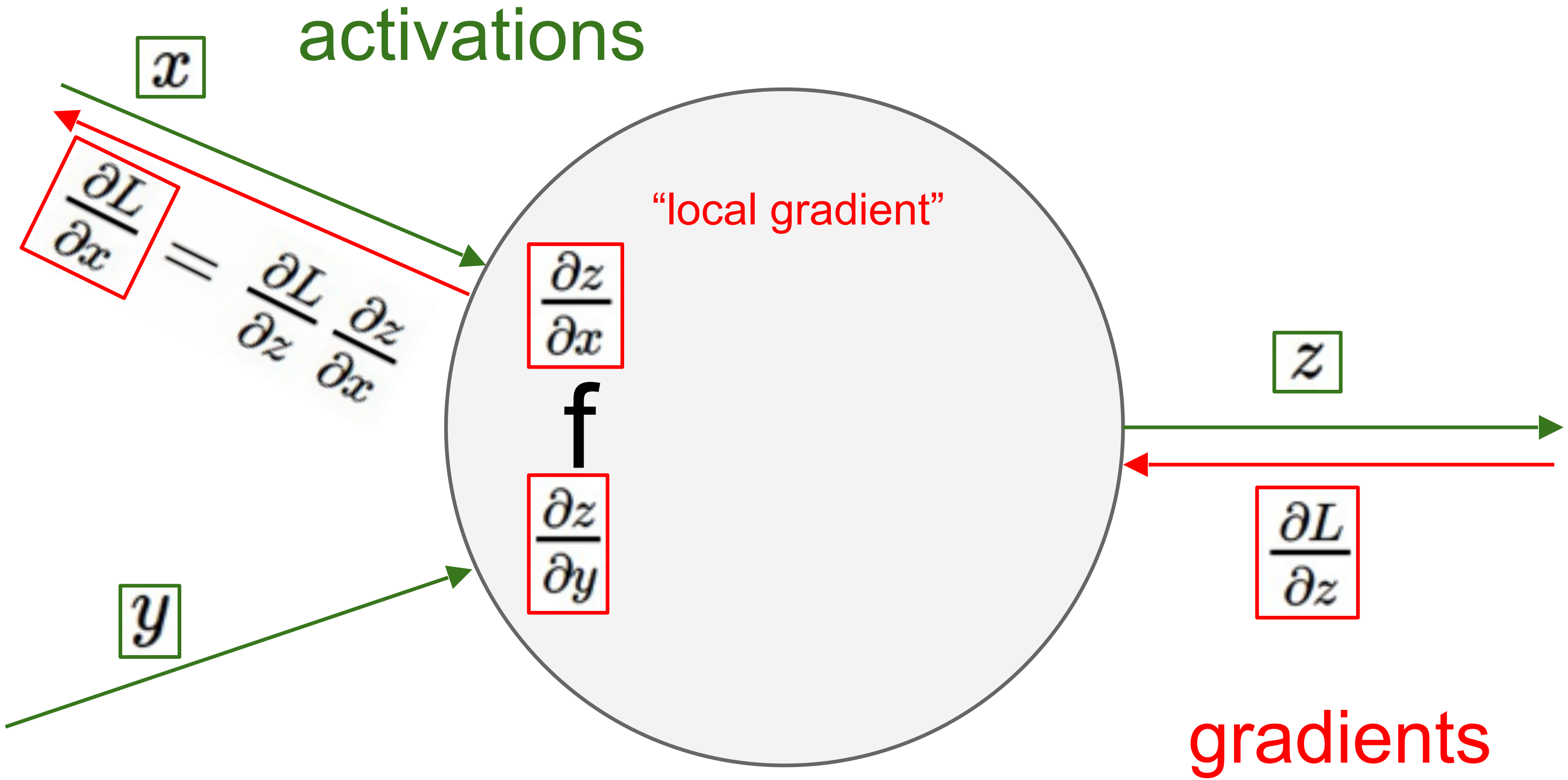
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$

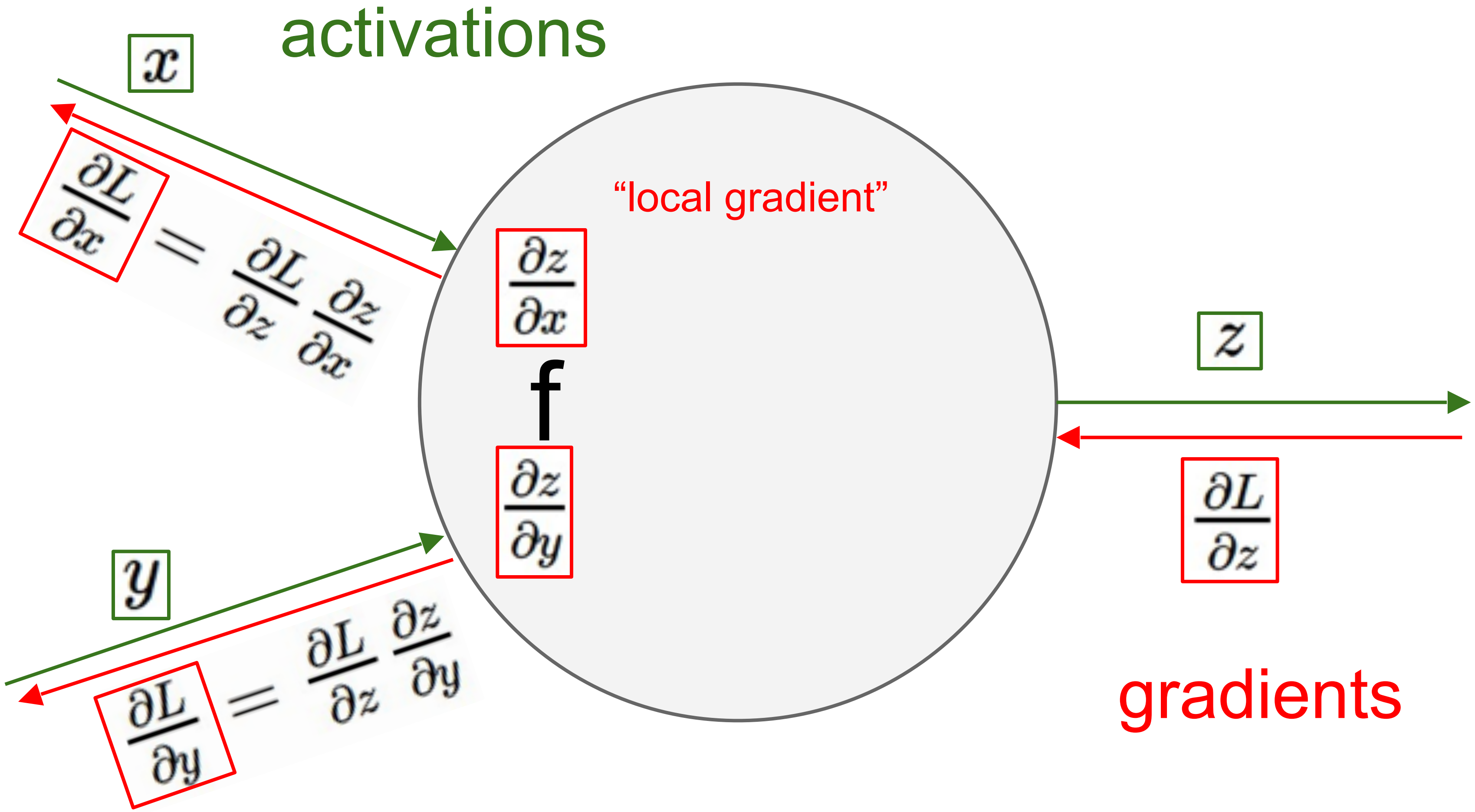


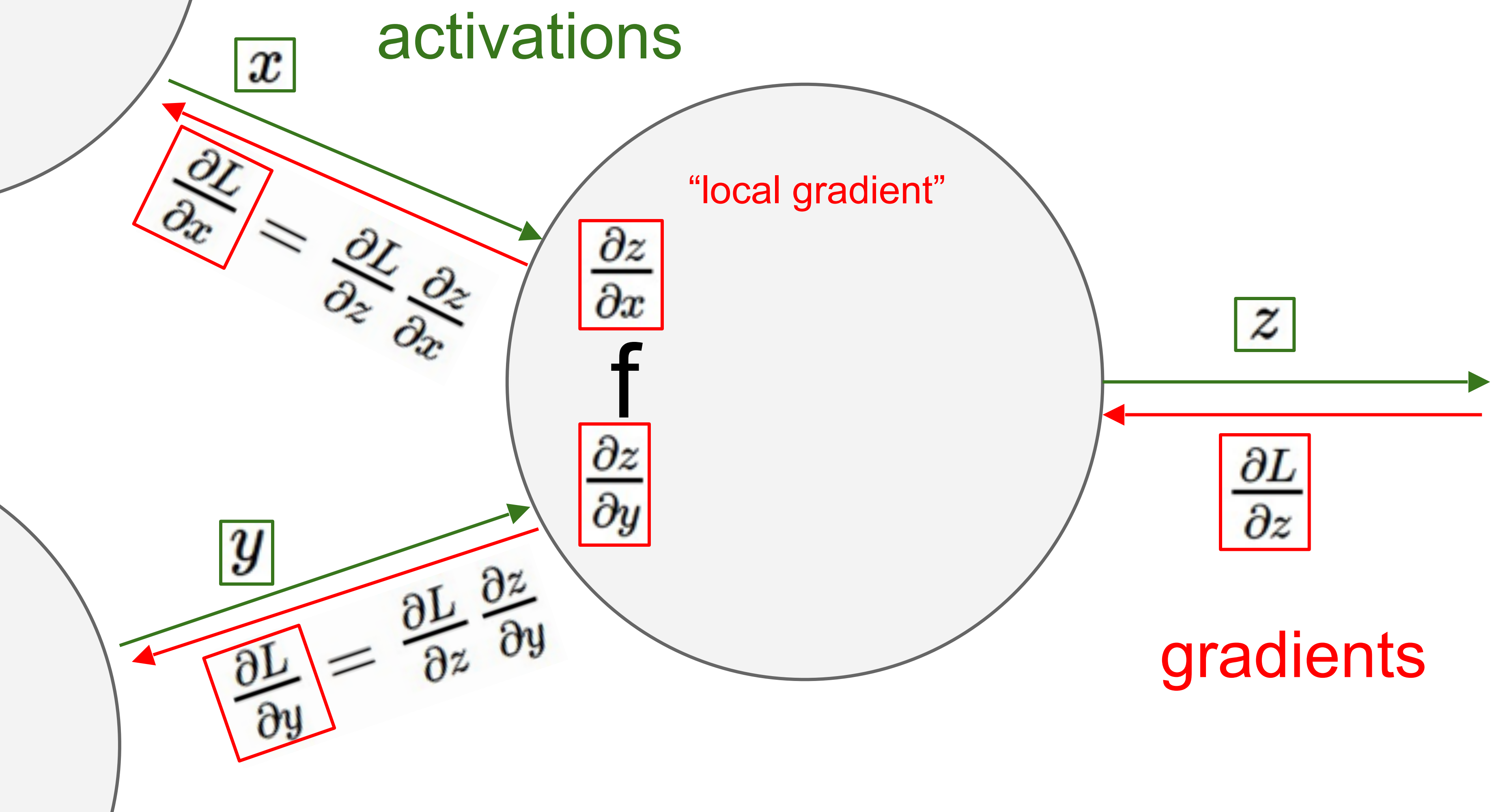
* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n


activations
x
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
y
z


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

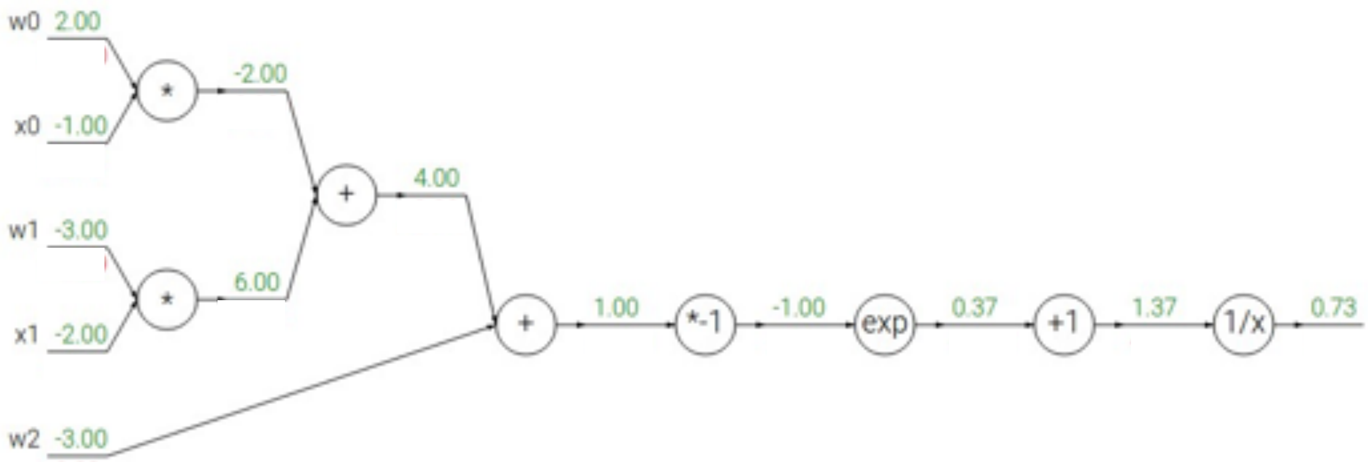

activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
y
z
$\frac{\partial L}{\partial z}$
gradients


activations
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


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

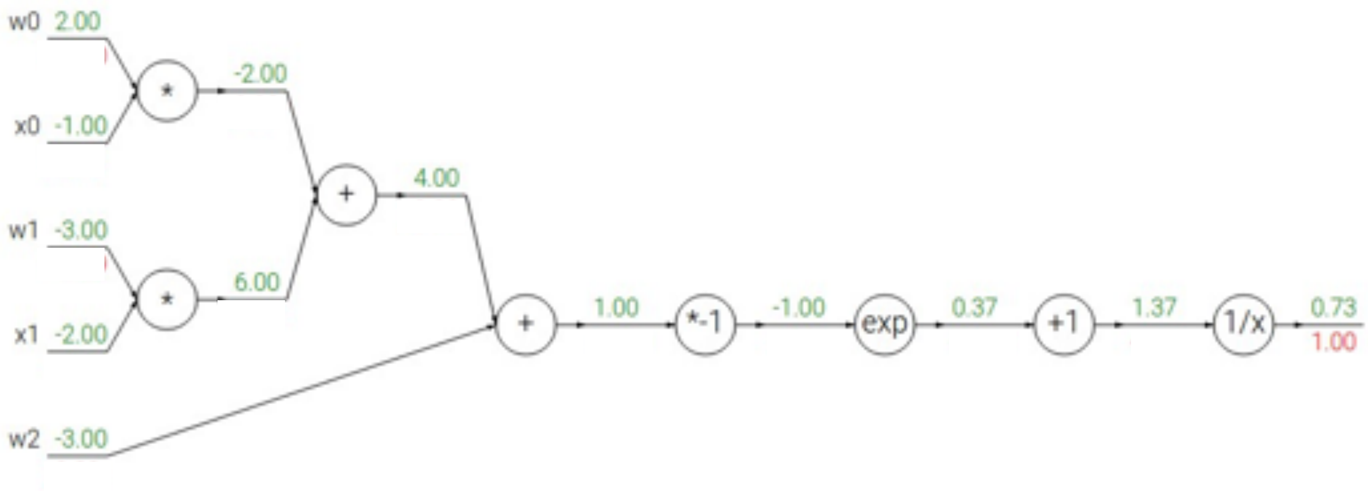
Another example:

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



Another example:

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



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

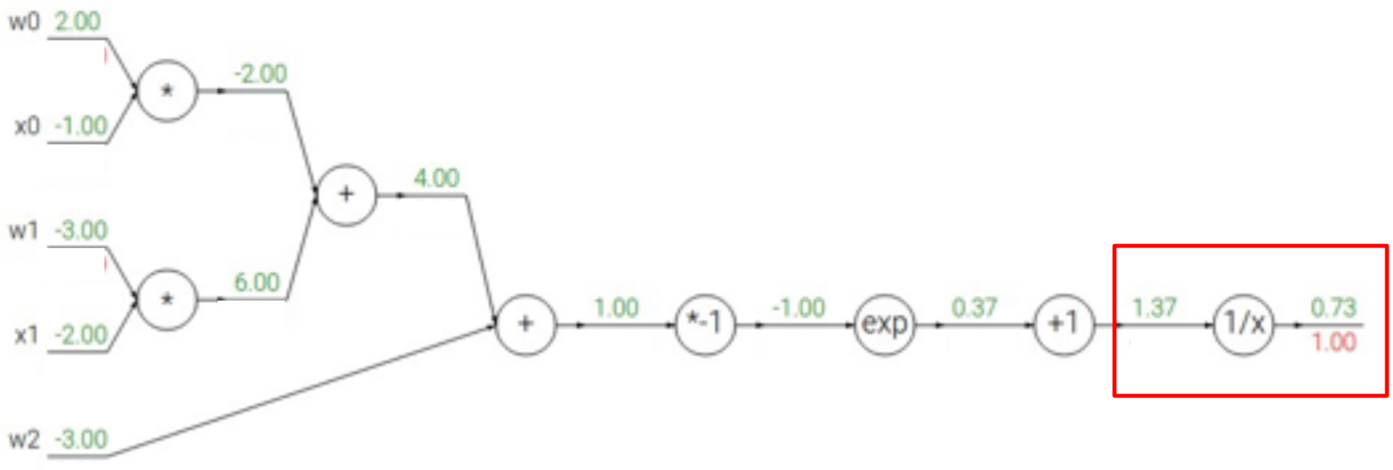
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



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

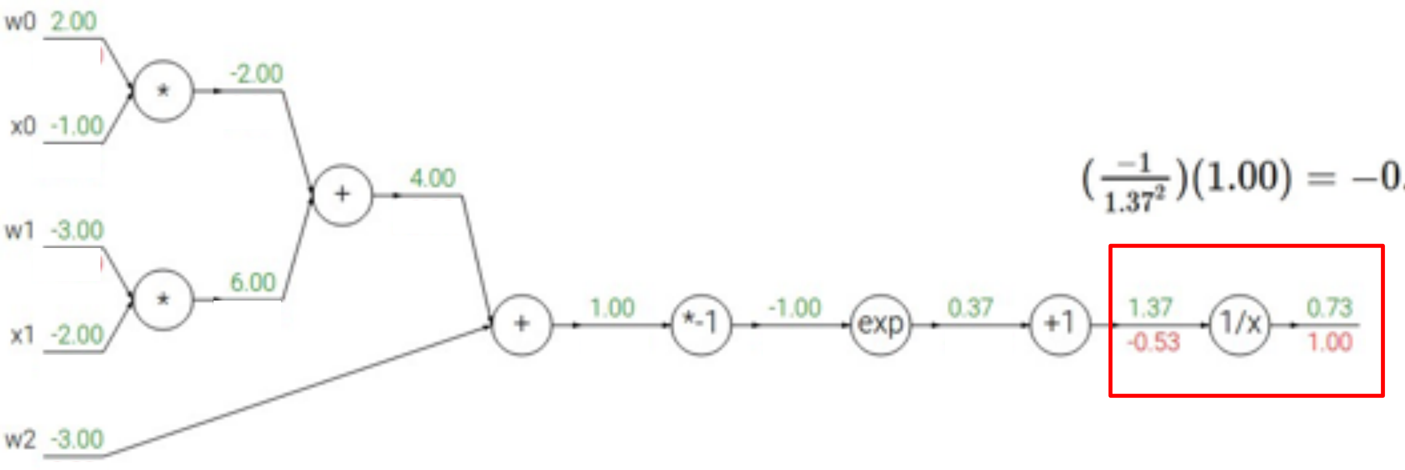
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



$$\left(\frac{-1}{1.37^2}\right)(1.00) = -0.53$$

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

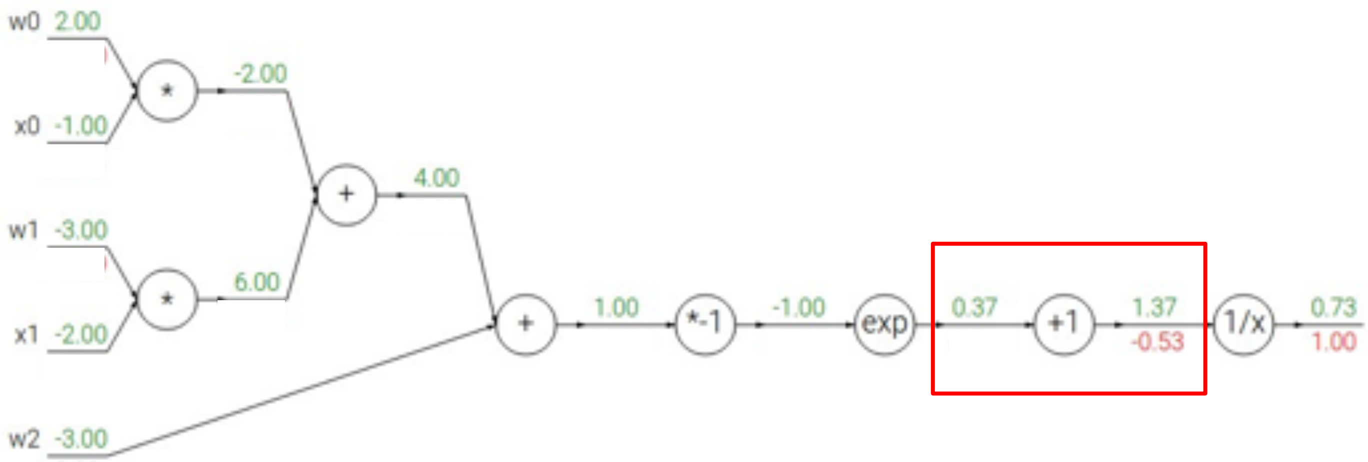
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



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

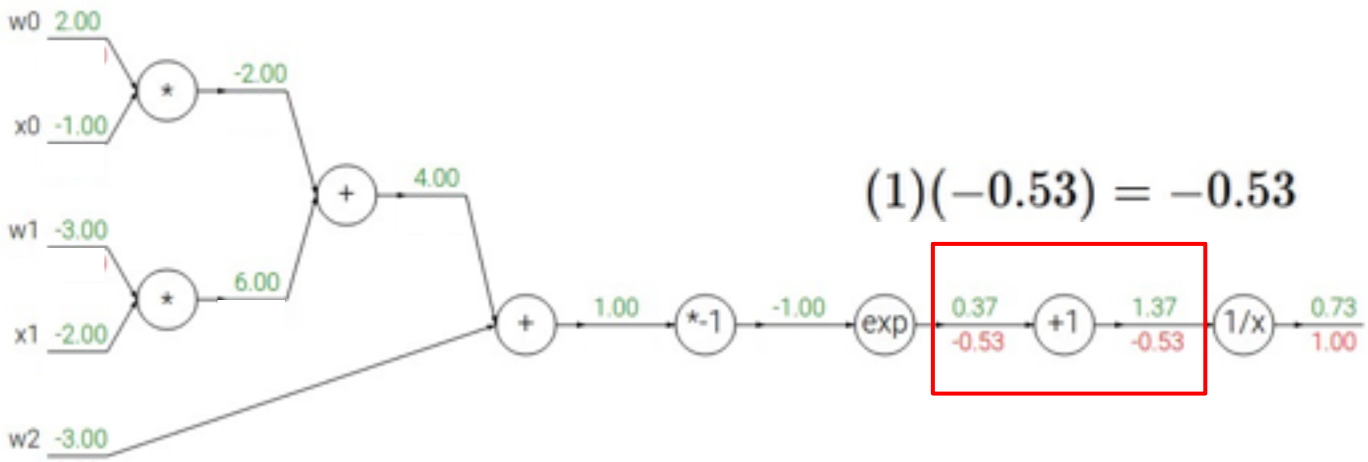
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



$$(1)(-0.53) = -0.53$$

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

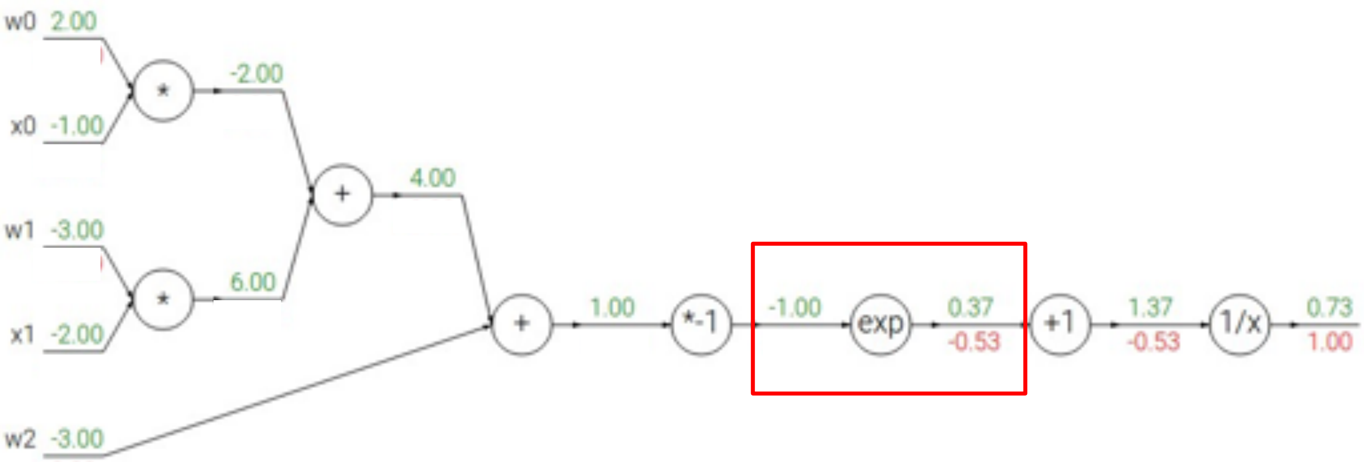
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



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

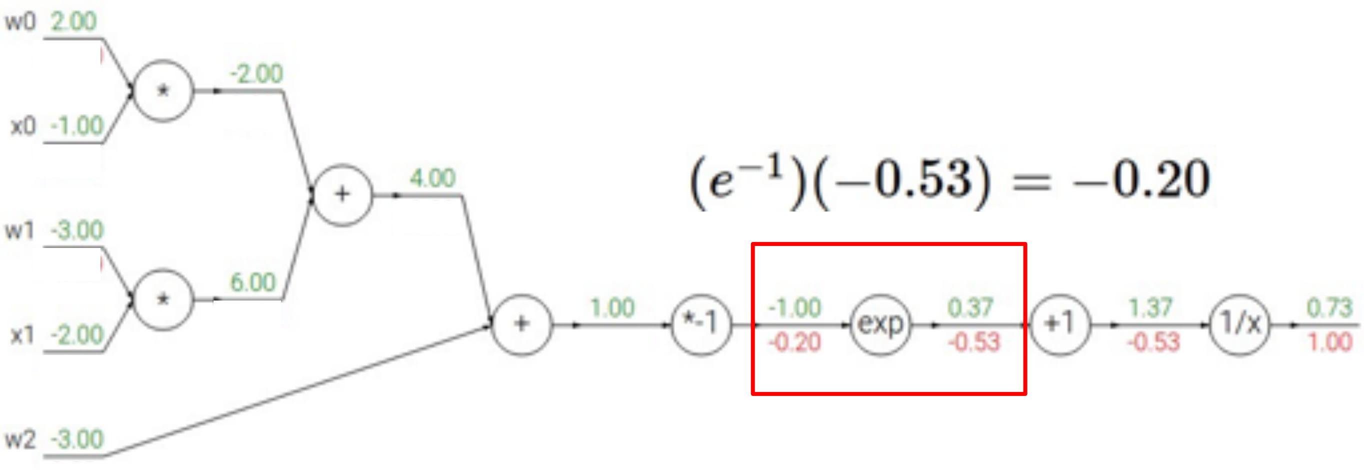
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



$$(e^{-1})(-0.53) = -0.20$$

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

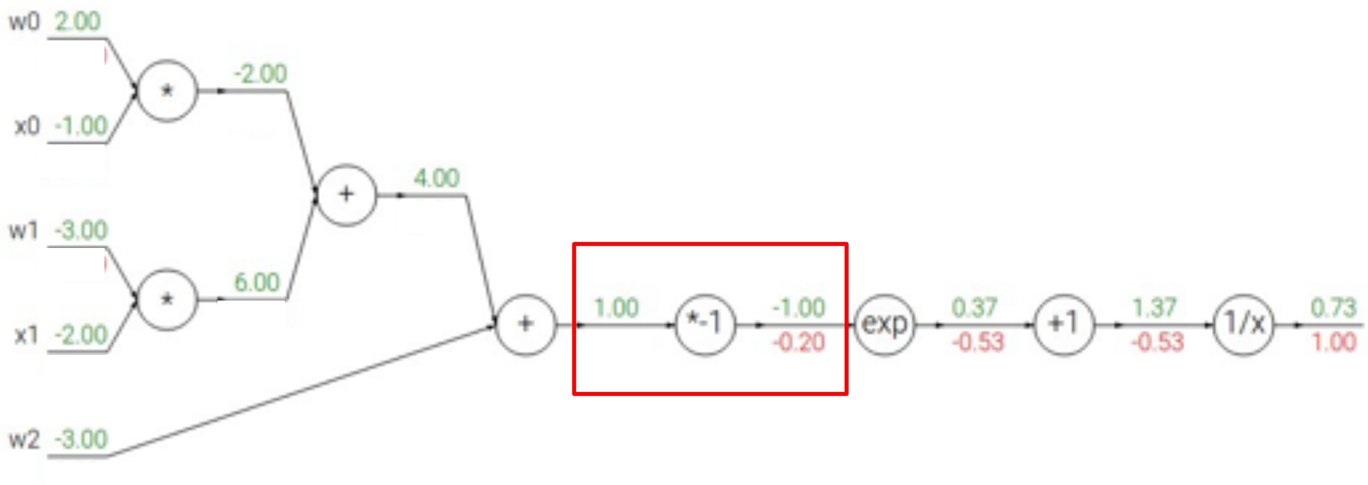
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



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

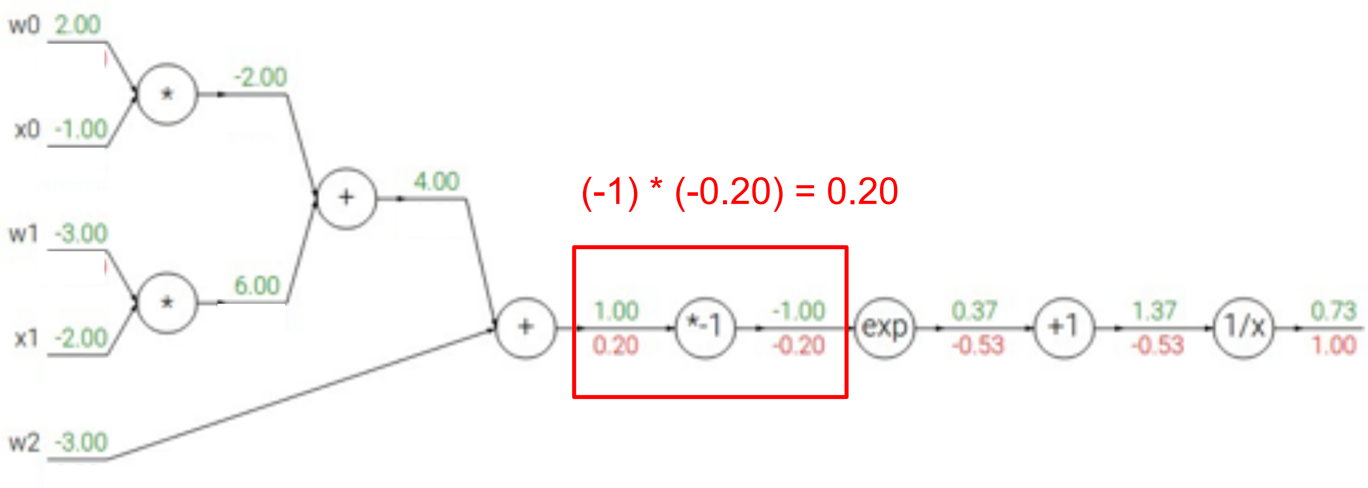
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

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

Another example:

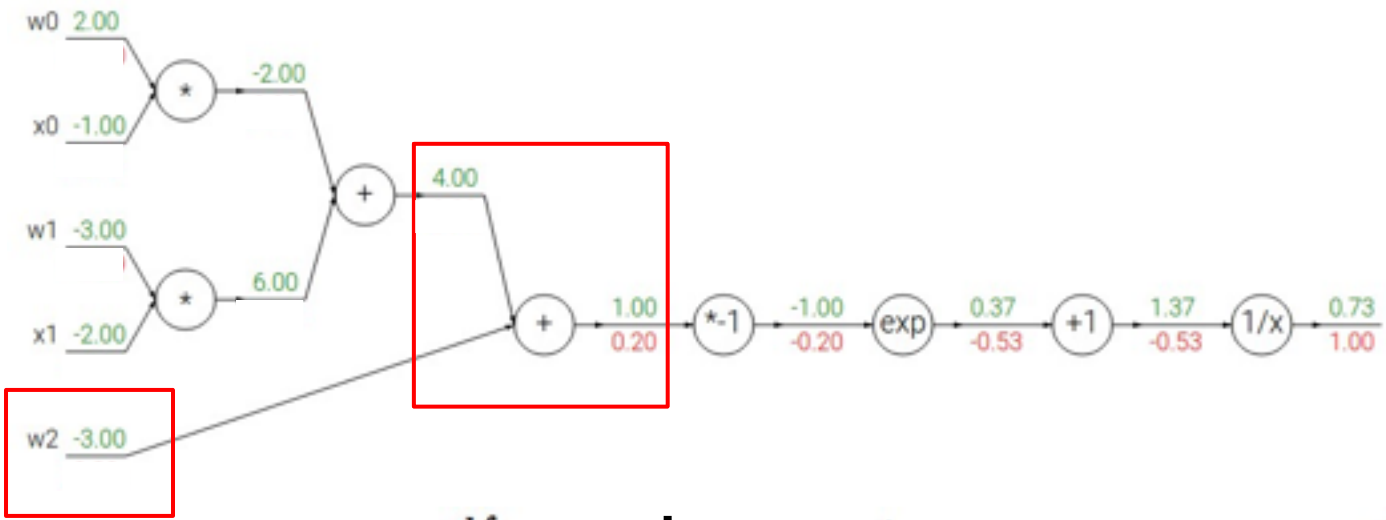
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



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

Another example:

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



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

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

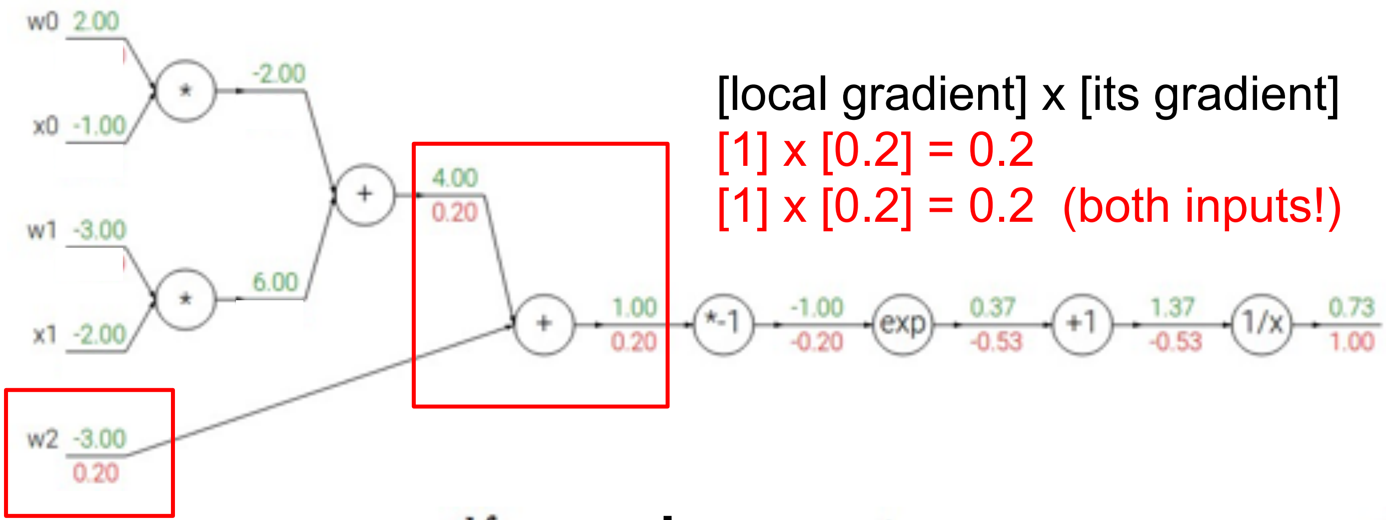
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

Another example:

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

[local gradient] x [its gradient]
[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)



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

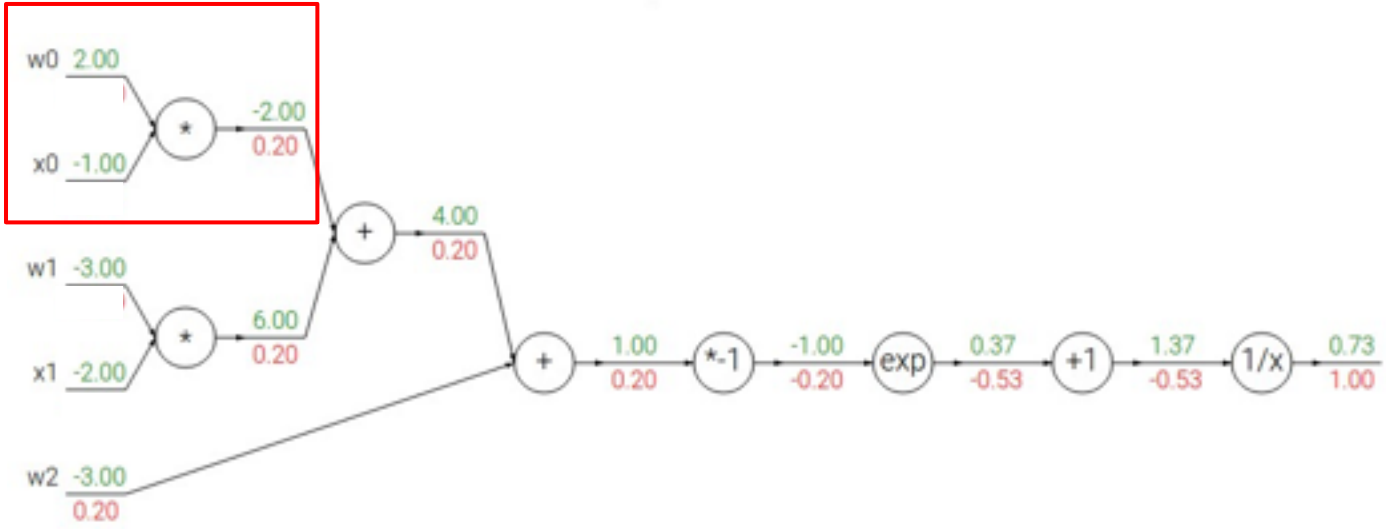
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



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

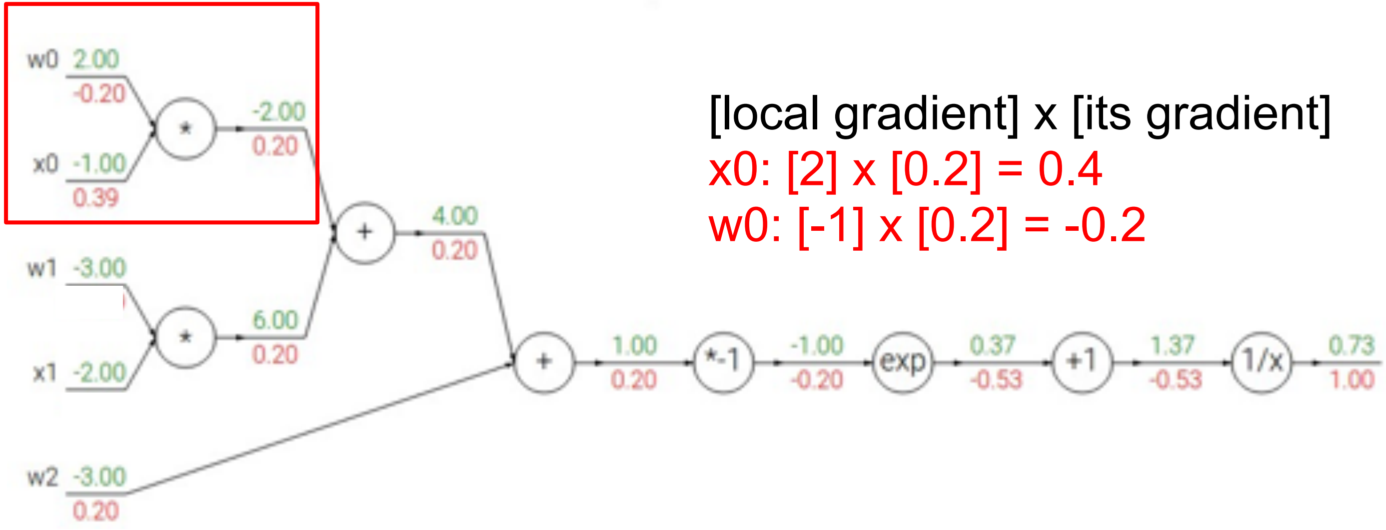
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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



[local gradient] x [its gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

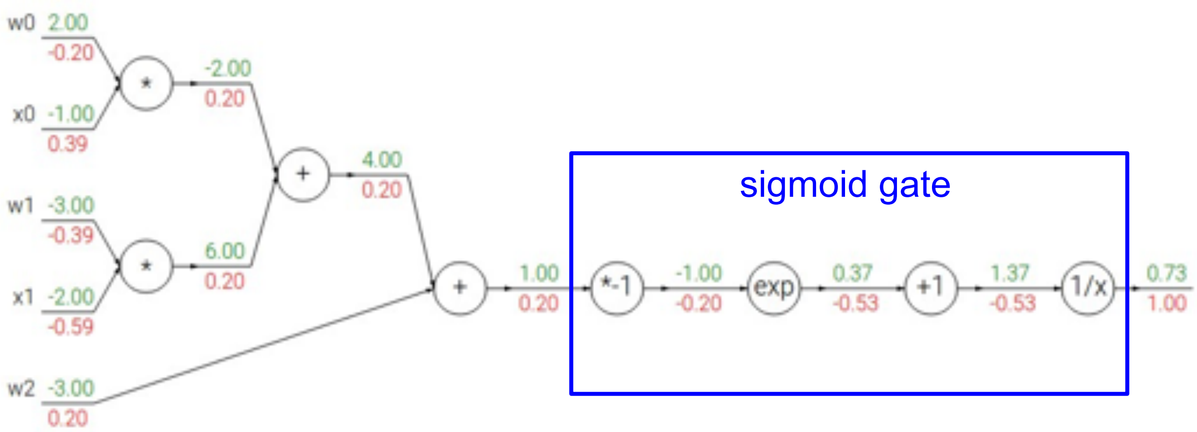
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

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

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

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

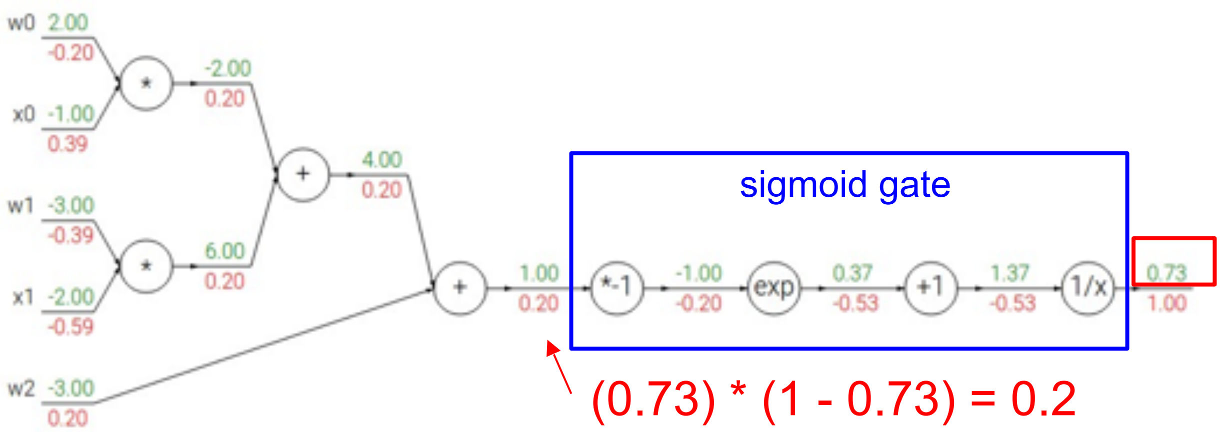
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



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

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

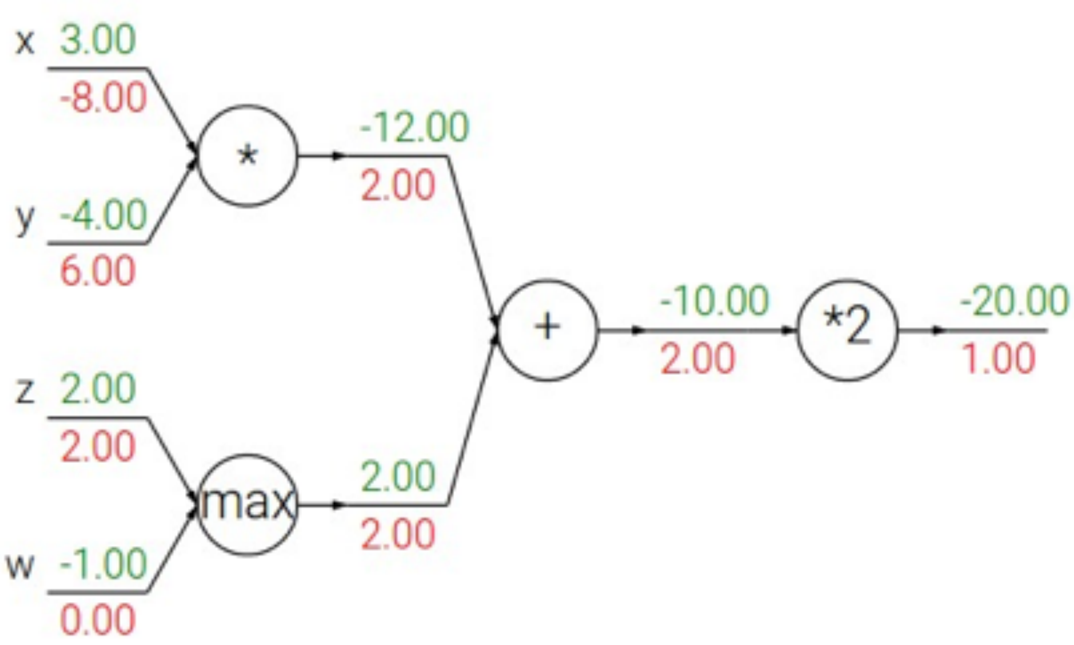
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



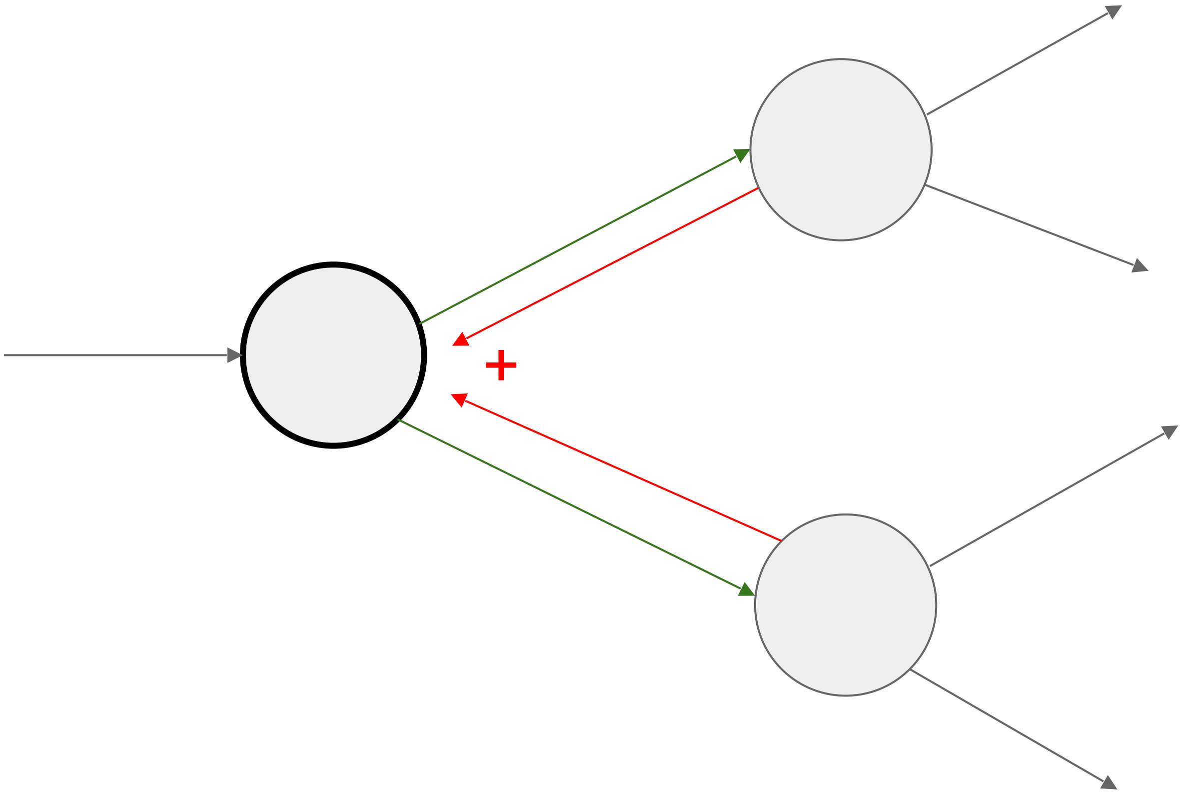
(0.73) * (1 - 0.73) = 0.2

Patterns in backward flow

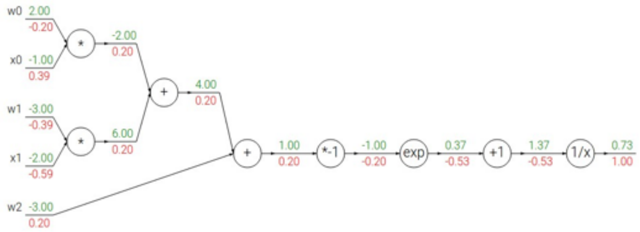
add gate: gradient distributor
max gate: gradient router
mul gate: gradient… “switcher”?



Gradients add at branches



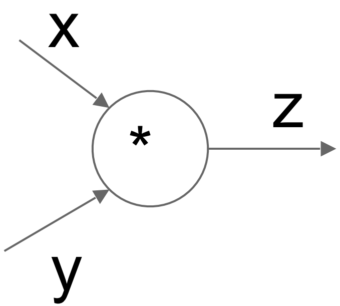
Implementation: forward/backward API



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

```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



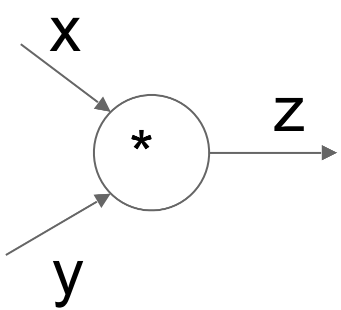
(x,y,z are scalars)

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

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

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

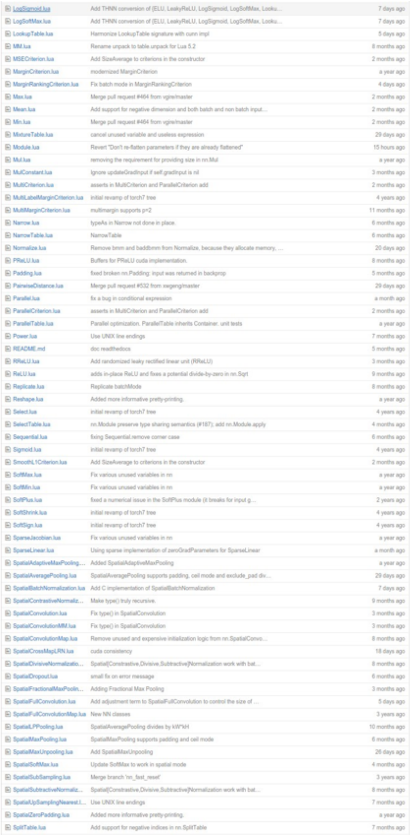
Implementation: forward/backward API

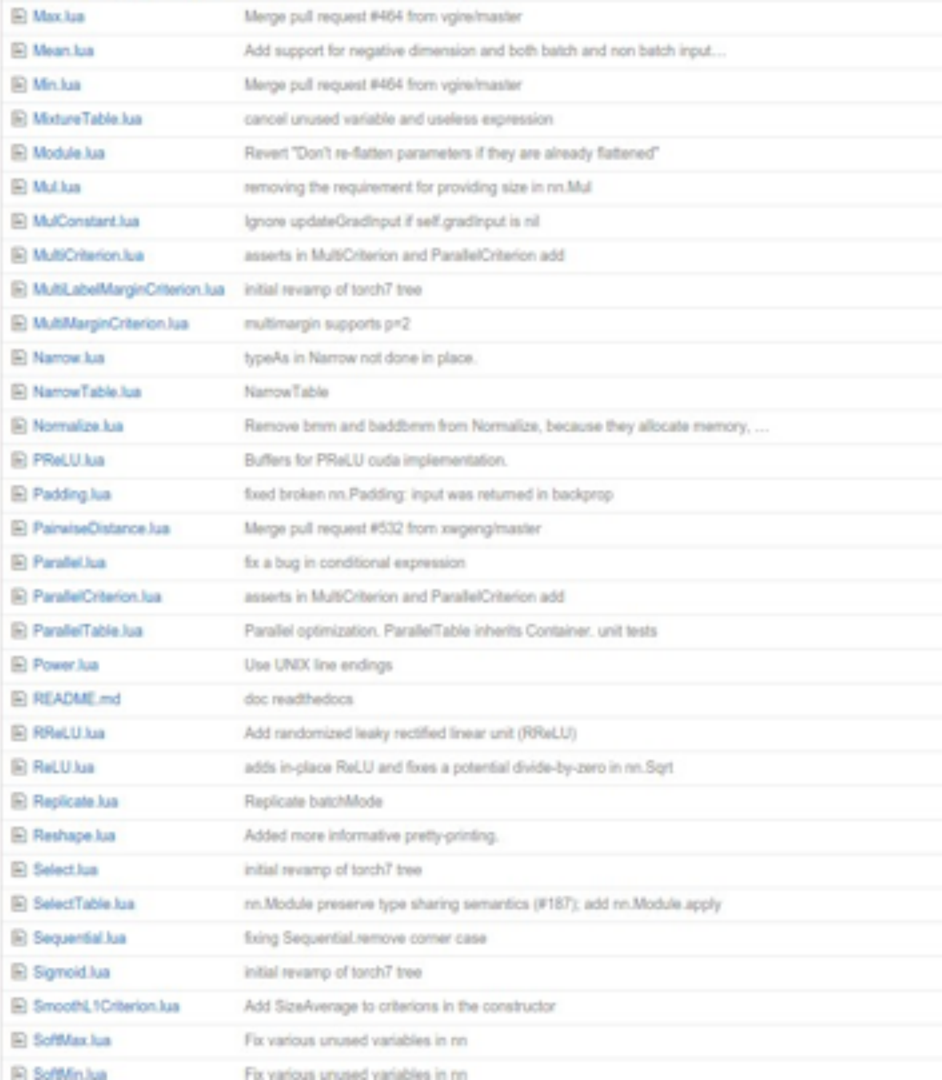



```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

(x,y,z are scalars)

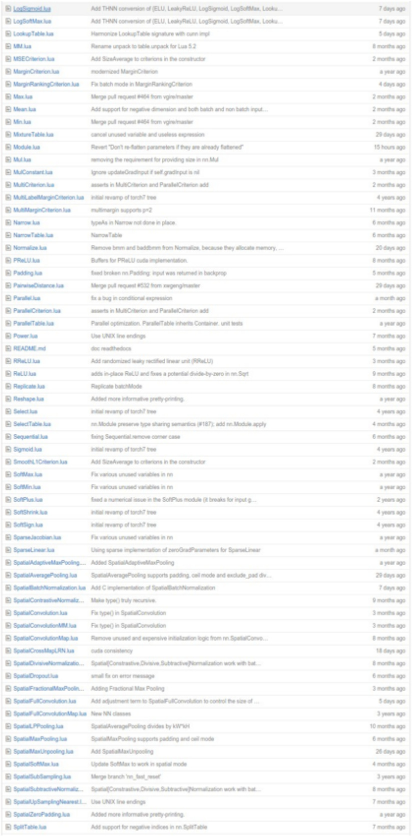
Example: Torch Layers





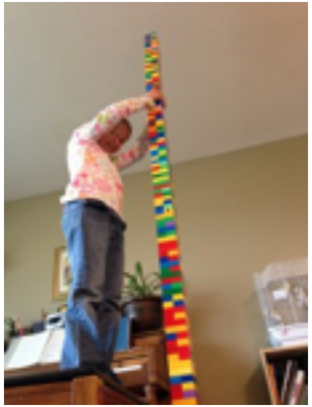
* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Example: Torch Layers



LogSigmoid.lua	Add THNN conversion of {ELU, LeakyReLU, LogSigmoid, LogSoftMax, Looku...	7 days ago
LogSoftMax.lua	Add THNN conversion of {ELU, LeakyReLU, LogSigmoid, LogSoftMax, Looku...	7 days ago
LookupTable.lua	Harmonize LookupTable signature with cunn impl	5 days ago
MM.lua	Rename unpack to table.unpack for Lua 5.2	8 months ago
MSECriterion.lua	Add SizeAverage to criterions in the constructor	2 months ago
MarginCriterion.lua	modernized MarginCriterion	a year ago
MarginRankingCriterion.lua	Fix batch mode in MarginRankingCriterion	4 days ago
Max.lua	Merge pull request #464 from vgire/master	2 months ago
Mean.lua	Add support for negative dimension and both batch and non batch input...	2 months ago
Min.lua	Merge pull request #464 from vgire/master	2 months ago
MixtureTable.lua	cancel unused variable and useless expression	29 days ago
Module.lua	Revert "Don't re-flatten parameters if they are already flattened"	15 hours ago
Mul.lua	removing the requirement for providing size in nn.Mul	a year ago
MulConstant.lua	Ignore updateGradInput if self.gradInput is nil	3 months ago
MultiCriterion.lua	asserts in MultiCriterion and ParallelCriterion add	2 months ago
MultiLabelMarginCriterion.lua	initial revamp of torch7 tree	4 years ago
MultiMarginCriterion.lua	multimargin supports p=2	11 months ago
Narrow.lua	typeAs in Narrow not done in place.	6 months ago
NarrowTable.lua	NarrowTable	6 months ago
Normalize.lua	Remove bmm and baddbmm from Normalize, because they allocate memory, ...	20 days ago
PReLU.lua	Buffers for PReLU cuda implementation.	8 months ago
Padding.lua	fixed broken nn.Padding: input was returned in backprop	5 months ago
PairwiseDistance.lua	Merge pull request #532 from xwgeng/master	29 days ago
Parallel.lua	fix a bug in conditional expression	a month ago
ParallelCriterion.lua	asserts in MultiCriterion and ParallelCriterion add	2 months ago
ParallelTable.lua	Parallel optimization. ParallelTable inherits Container. unit tests	a year ago
Power.lua	Use UNIX line endings	7 months ago
README.md	doc readthedocs	5 months ago
RReLU.lua	Add randomized leaky rectified linear unit (RReLU)	3 months ago
ReLU.lua	adds in-place ReLU and fixes a potential divide-by-zero in nn.Sqrt	9 months ago
Replicate.lua	Replicate batchMode	8 months ago
Reshape.lua	Added more informative pretty-printing.	a year ago
Select.lua	initial revamp of torch7 tree	4 years ago
SelectTable.lua	nn.Module preserve type sharing semantics (#187); add nn.Module.apply	4 months ago
Sequential.lua	fixing Sequential.remove corner case	6 months ago
Sigmoid.lua	initial revamp of torch7 tree	4 years ago
SmoothL1Criterion.lua	Add SizeAverage to criterions in the constructor	2 months ago
SoftMax.lua	Fix various unused variables in nn	a year ago
SoftMin.lua	Fix various unused variables in nn	a year ago
SoftPlus.lua	fixed a numerical issue in the SoftPlus module (it breaks for input g...	2 years ago
SoftShrink.lua	initial revamp of torch7 tree	4 years ago
SoftSign.lua	initial revamp of torch7 tree	4 years ago
SparseJacobian.lua	Fix various unused variables in nn	a year ago
SparseLinear.lua	Using sparse implementation of zeroGradParameters for SparseLinear	a month ago
SpatialAdaptiveMaxPooling....	Added SpatialAdaptiveMaxPooling	a year ago
SpatialAveragePooling.lua	SpatialAveragePooling supports padding, ceil mode and exclude_pad div...	29 days ago
SpatialBatchNormalization.lua	Add C implementation of SpatialBatchNormalization	7 days ago
SpatialContrastiveNormaliz...	Make type() truly recursive.	9 months ago
SpatialConvolution.lua	Fix type() in SpatialConvolution	3 months ago
SpatialConvolutionMM.lua	Fix type() in SpatialConvolution	3 months ago
SpatialConvolutionMap.lua	Remove unused and expensive initialization logic from nn.SpatialConvo...	8 months ago
SpatialCrossMapLRN.lua	cuda consistency	18 days ago
SpatialDivisiveNormalizatio...	Spatial{Constrastive,Divisive,Subtractive}Normalization work with bat...	8 months ago
SpatialDropout.lua	small fix on error message	6 months ago
SpatialFractionalMaxPoolin...	Adding Fractional Max Pooling	3 months ago
SpatialFullConvolution.lua	Add adjustment term to SpatialFullConvolution to control the size of ...	5 days ago
SpatialFullConvolutionMap.lua	New NN classes	3 years ago
SpatialLPPooling.lua	SpatialAveragePooling divides by kW*kH	10 months ago
SpatialMaxPooling.lua	SpatialMaxPooling supports padding and ceil mode	6 months ago
SpatialMaxUnpooling.lua	Add SpatialMaxUnpooling	26 days ago
SpatialSoftMax.lua	Update SoftMax to work in spatial mode	4 months ago
SpatialSubSampling.lua	Merge branch 'nn_fast_reset'	3 years ago
SpatialSubtractiveNormaliz...	Spatial{Constrastive,Divisive,Subtractive}Normalization work with bat...	8 months ago
SpatialUpSamplingNearest.l...	Use UNIX line endings	7 months ago
SpatialZeroPadding.lua	Added more informative pretty-printing.	a year ago
SplitTable.lua	Add support for negative indices in nn.SplitTable	7 months ago

=



Example: Torch MulConstant

```lua
local MulConstant, parent = torch.class('nn.MulConstant', 'nn.Module')

function MulConstant:__init(constant_scalar,ip)
  parent.__init(self)
  assert(type(constant_scalar) == 'number', 'input is not scalar!')
  self.constant_scalar = constant_scalar

  -- default for inplace is false
   self.inplace = ip or false
   if (ip and type(ip) ~= 'boolean') then
      error('in-place flag must be boolean')
   end
end

function MulConstant:updateOutput(input)
  if self.inplace then
    input:mul(self.constant_scalar)
    self.output = input
  else
    self.output:resizeAs(input)
    self.output:copy(input)
    self.output:mul(self.constant_scalar)
  end
  return self.output
end

function MulConstant:updateGradInput(input, gradOutput)
  if self.gradInput then
    if self.inplace then
      gradOutput:mul(self.constant_scalar)
      self.gradInput = gradOutput
      -- restore previous input value
      input:div(self.constant_scalar)
    else
      self.gradInput:resizeAs(gradOutput)
      self.gradInput:copy(gradOutput)
      self.gradInput:mul(self.constant_scalar)
    end
    return self.gradInput
  end
end
```

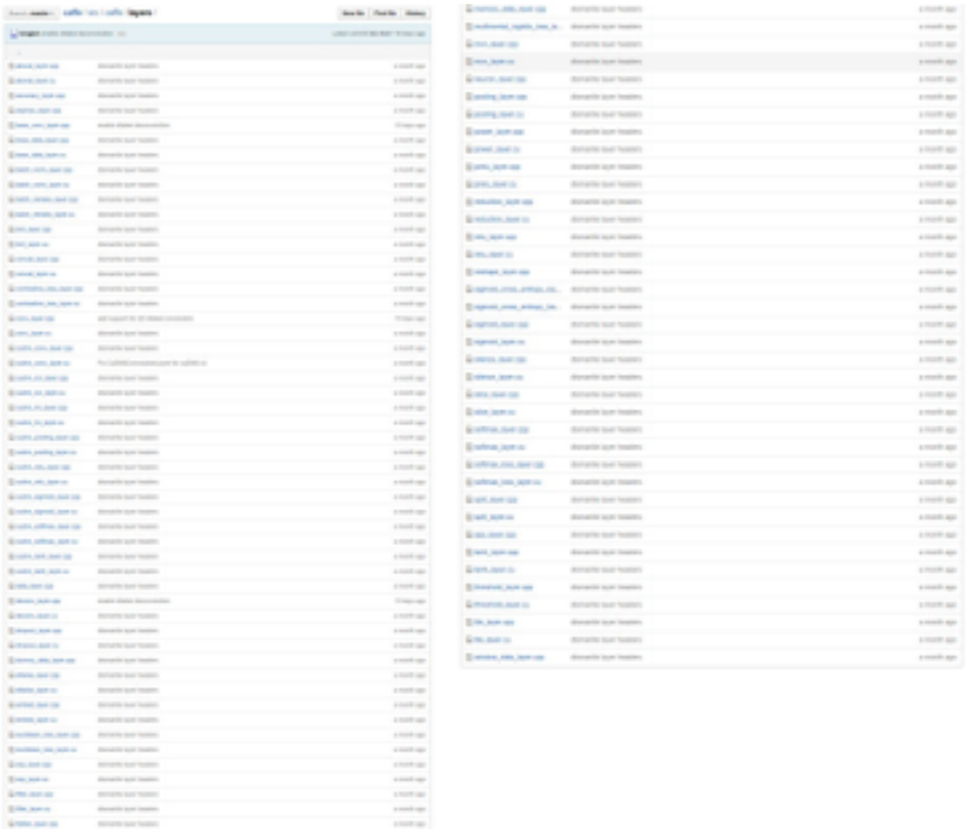
$$f(X) = aX$$

initialization

forward()

backward()

Example: Caffe Layers



Caffe Sigmoid Layer

```cpp
#include <cmath>
#include <vector>

#include "caffe/layers/sigmoid_layer.hpp"

namespace caffe {

template <typename Dtype>
inline Dtype sigmoid(Dtype x) {
  return 1. / (1. + exp(-x));
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
    const vector<Blob<Dtype>*>& top) {
  const Dtype* bottom_data = bottom[0]->cpu_data();
  Dtype* top_data = top[0]->mutable_cpu_data();
  const int count = bottom[0]->count();
  for (int i = 0; i < count; ++i) {
    top_data[i] = sigmoid(bottom_data[i]);
  }
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down,
    const vector<Blob<Dtype>*>& bottom) {
  if (propagate_down[0]) {
    const Dtype* top_data = top[0]->cpu_data();
    const Dtype* top_diff = top[0]->cpu_diff();
    Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
    const int count = bottom[0]->count();
    for (int i = 0; i < count; ++i) {
      const Dtype sigmoid_x = top_data[i];
      bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
    }
  }
}

#ifdef CPU_ONLY
STUB_GPU(SigmoidLayer);
#endif

INSTANTIATE_CLASS(SigmoidLayer);


}  // namespace caffe
```

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

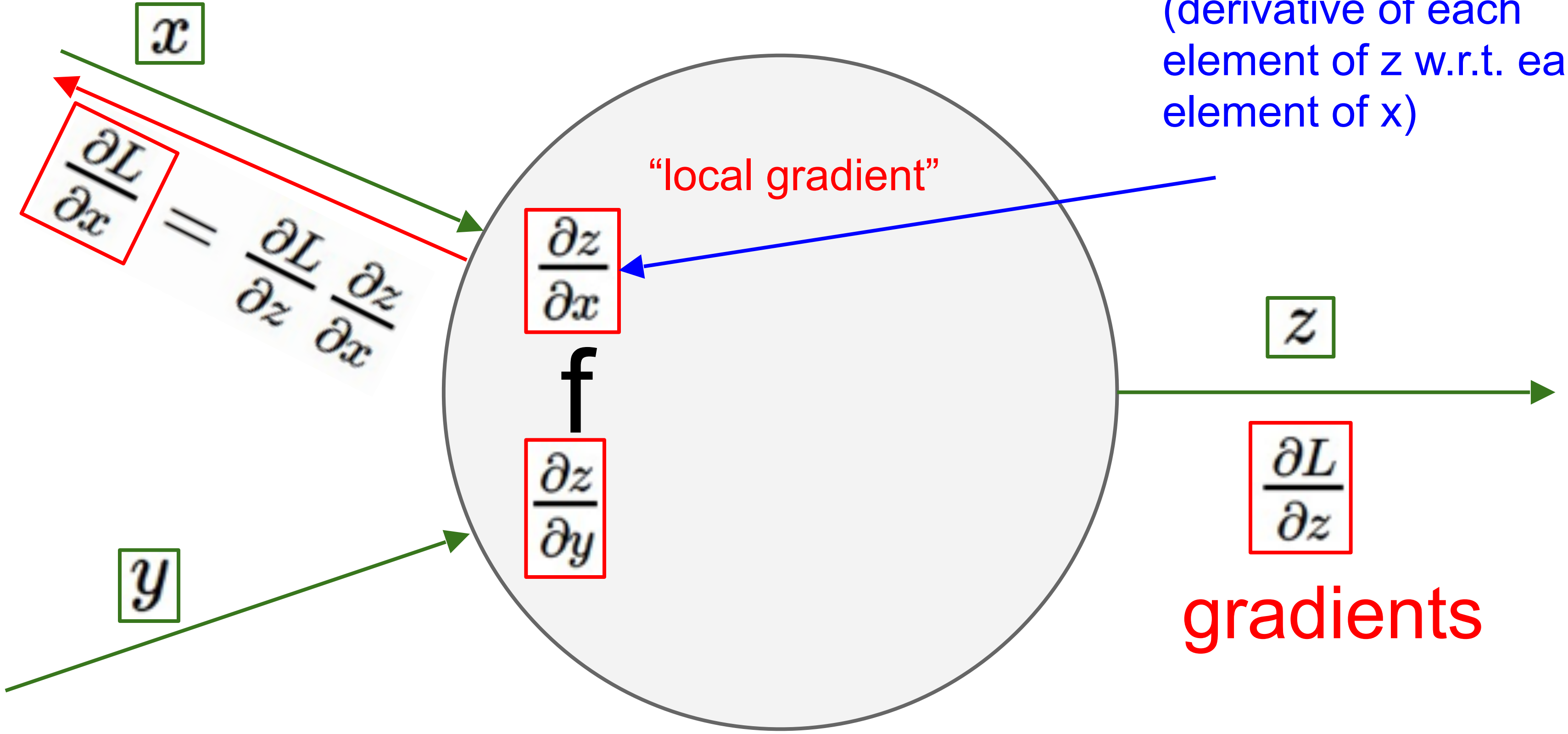
$\boxed{(1-\sigma(x))\sigma(x)}$ *top_diff (chain rule)

```cpp
#include <vector>
#include <random>
#include <iterator>
#include "caffe/layers/shuffle_layer.hpp"
#include "caffe/util/math_functions.hpp"

namespace caffe {

  template <typename Dtype>
  void Shuffle(Dtype* bottom_data, Dtype* top_data, const int item_size,
          const bool forward, const int* shuffle_order, const int count) {
    // data_shape is expected to be the shape (count, M) of the blob
    // data in bottom_data and top_data
    for (int i = 0; i < count; ++i) {
      for (int j = 0; j < item_size; ++j) {
        if (forward) {
      top_data[i*item_size+j] = bottom_data[i*item_size+shuffle_order[j]];
        } else {
      bottom_data[i*item_size+shuffle_order[j]] = top_data[i*item_size+j];
        }
      }
    }
  }

  template <typename Dtype>
  void ShuffleLayer<Dtype>::LayerSetUp(const vector<Blob<Dtype>*>& bottom,
            const vector<Blob<Dtype>*>& top) {
    // Check there is only one bottom layer
    CHECK_EQ(bottom.size(), 1);
    // TODO: extend functionality to > 2-D blobs, but for now only 2D works
    // CHECK_EQ(bottom[0]->shape().size(), 2);

    shuffle_seed_ = rand();
    // calculate count of each item in batch
    batch_item_size_ = bottom[0]->count(1, bottom[0]->CanonicalAxisIndex(-1));

    vector<int> shuffle_order;
    // Make a vector of ordered inds
    for (int i=0; i<bottom[0]->shape(1); i++) shuffle_order.push_back(i);
    // Mersenne twister initialized with input seed
    std::mt19937 gen(shuffle_seed_);
    std::shuffle(shuffle_order.begin(), shuffle_order.end(), gen);

    // copy randomized shuffle order to layer member variable shuffle_order_
    shuffle_order_.Reshape(shuffle_order.size(), 1, 1, 1);

    for (int i = 0; i < shuffle_order.size(); i++) {
      shuffle_order_.mutable_cpu_data()[shuffle_order_.offset(i)] = shuffle_order[i];
    }
```

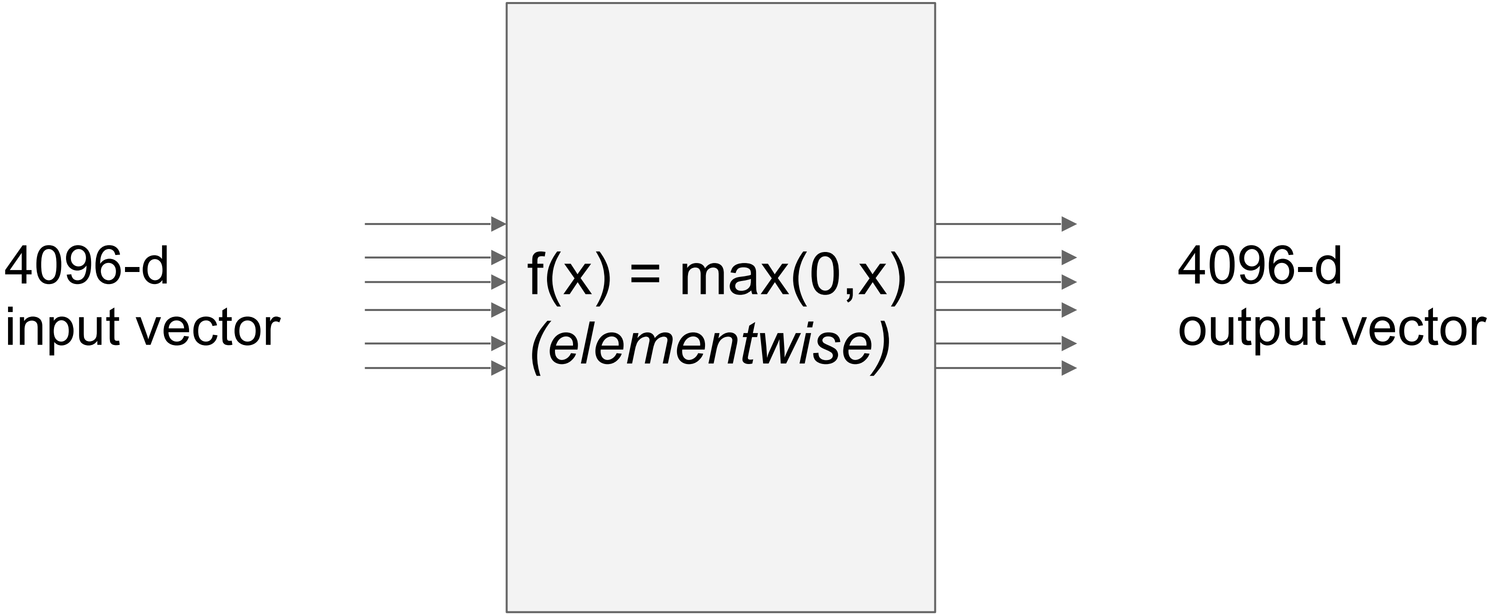
```cpp
  template <typename Dtype>
  void ShuffleLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
          const vector<Blob<Dtype>*>& top) {
    const int count = top[0]->num();
    std::cout << bottom[0]->shape(0) << std::endl;
    std::cout << bottom[0]->shape(1) << std::endl;
    vector<int> orig_shape = bottom[0]->shape();
    vector<int> new_shape; new_shape.push_back(count); new_shape.push_back(batch_item_size_);
    bottom[0]->Reshape(new_shape);
    top[0]->Reshape(new_shape);
    Dtype* bottom_data = bottom[0]->mutable_cpu_data();
    Dtype* top_data = top[0]->mutable_cpu_data();
    const int* shuffle_order = shuffle_order_.cpu_data();

    bool forward = true;
    Shuffle(bottom_data, top_data, batch_item_size_, forward, shuffle_order,
        count);
    top[0]->Reshape(orig_shape);
    bottom[0]->Reshape(orig_shape);
  }

  template <typename Dtype>
  void ShuffleLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
          const vector<bool>& propagate_down, const vector<Blob<Dtype>*>& bottom) {
    const int count = top[0]->num();
    vector<int> orig_shape = bottom[0]->shape();
    vector<int> new_shape; new_shape.push_back(count); new_shape.push_back(batch_item_size_);
    bottom[0]->Reshape(new_shape);
    top[0]->Reshape(new_shape);
    Dtype* bottom_data = bottom[0]->mutable_cpu_data();
    Dtype* top_data = top[0]->mutable_cpu_data();
    const int* shuffle_order = shuffle_order_.cpu_data();

    bool forward = false;
    Shuffle(bottom_data, top_data, batch_item_size_, forward, shuffle_order,
          count);
    bottom[0]->Reshape(orig_shape);
    top[0]->Reshape(orig_shape);
  }

  #ifdef CPU_ONLY
  STUB_GPU(ShuffleLayer);
  #endif

  INSTANTIATE_CLASS(ShuffleLayer);
  REGISTER_LAYER_CLASS(Shuffle);

} // namespace caffe
```

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

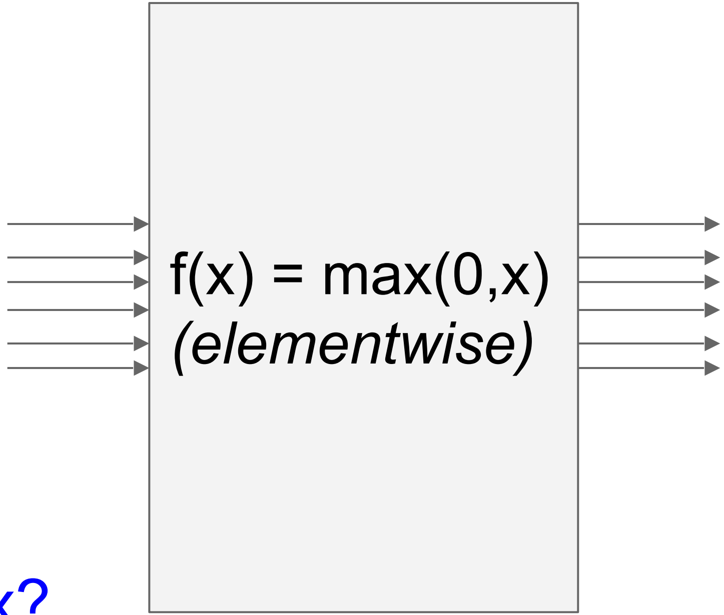


Vectorized operations

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

Jacobian matrix

4096-d
input vector



4096-d
output vector

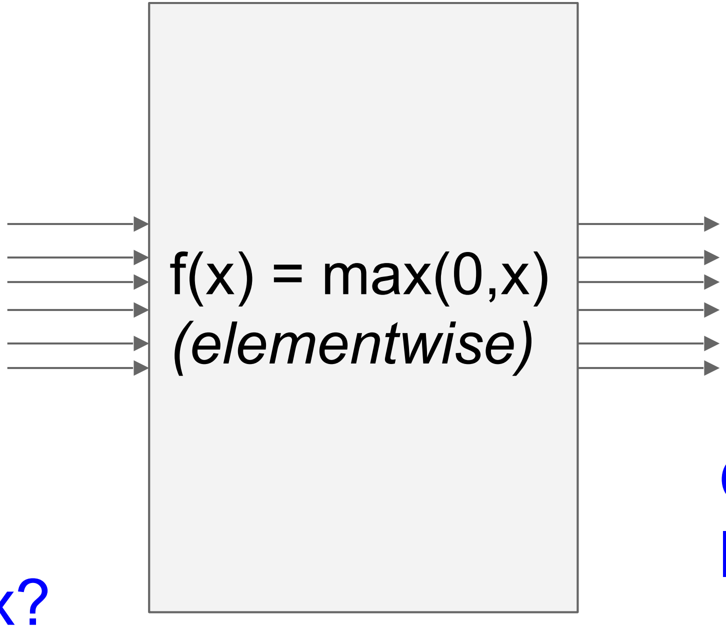
Q: what is the size of the Jacobian matrix?

Vectorized operations

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

Jacobian matrix

4096-d
input vector



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

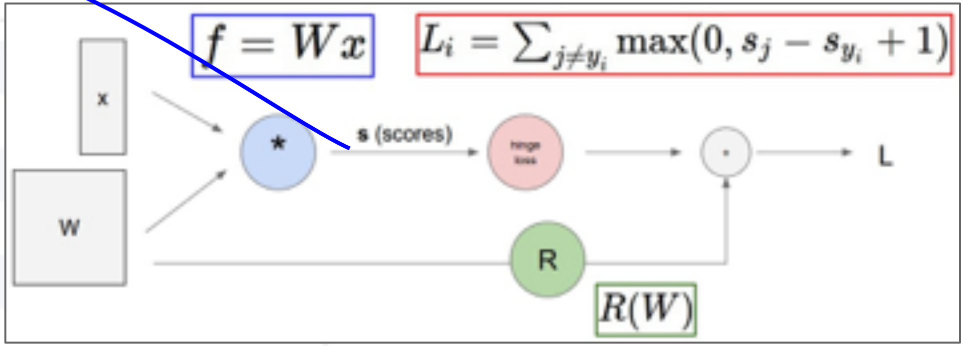
i.e. Jacobian would technically be a [409,600 x 409,600] matrix :\

Assignment: Writing SVM/Softmax
Stage your forward/backward computation!

margins

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**().
- **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 so far:

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

Neural Network so far:

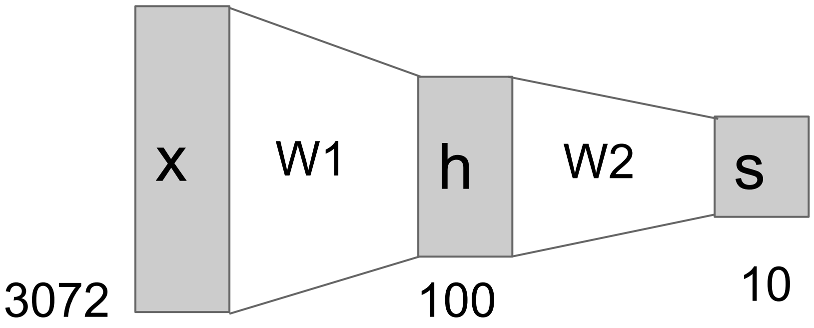
(**Before**) Linear score function: $f = Wx$

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

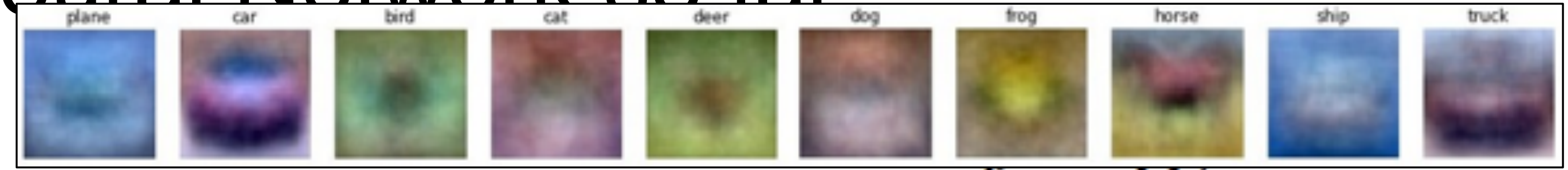
Neural Network so far:

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

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

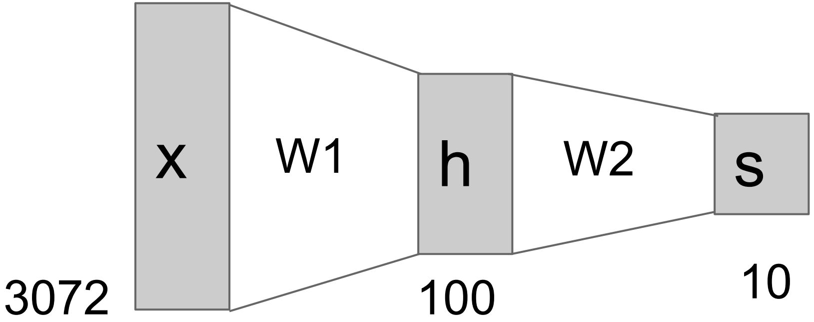


Neural Network so far:



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

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



* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Neural Network so far:

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

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

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

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

```
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in xrange(60000):
    l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
    l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
    l2_delta = (y - l2)*(l2*(1-l2))
    l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
    syn1 += l1.T.dot(l2_delta)
    syn0 += X.T.dot(l1_delta)
```

Forward pass

Backward pass

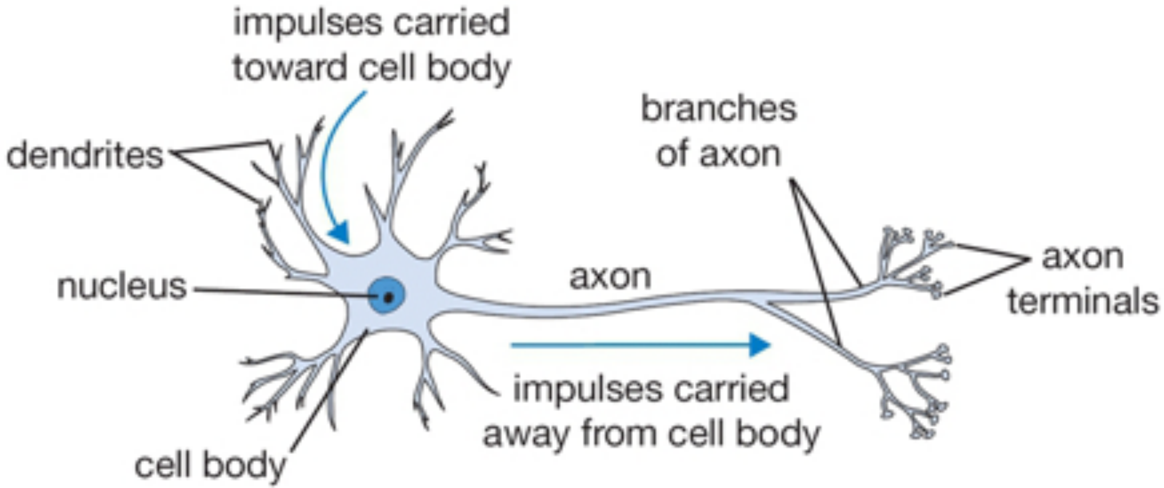
backprop of derivative

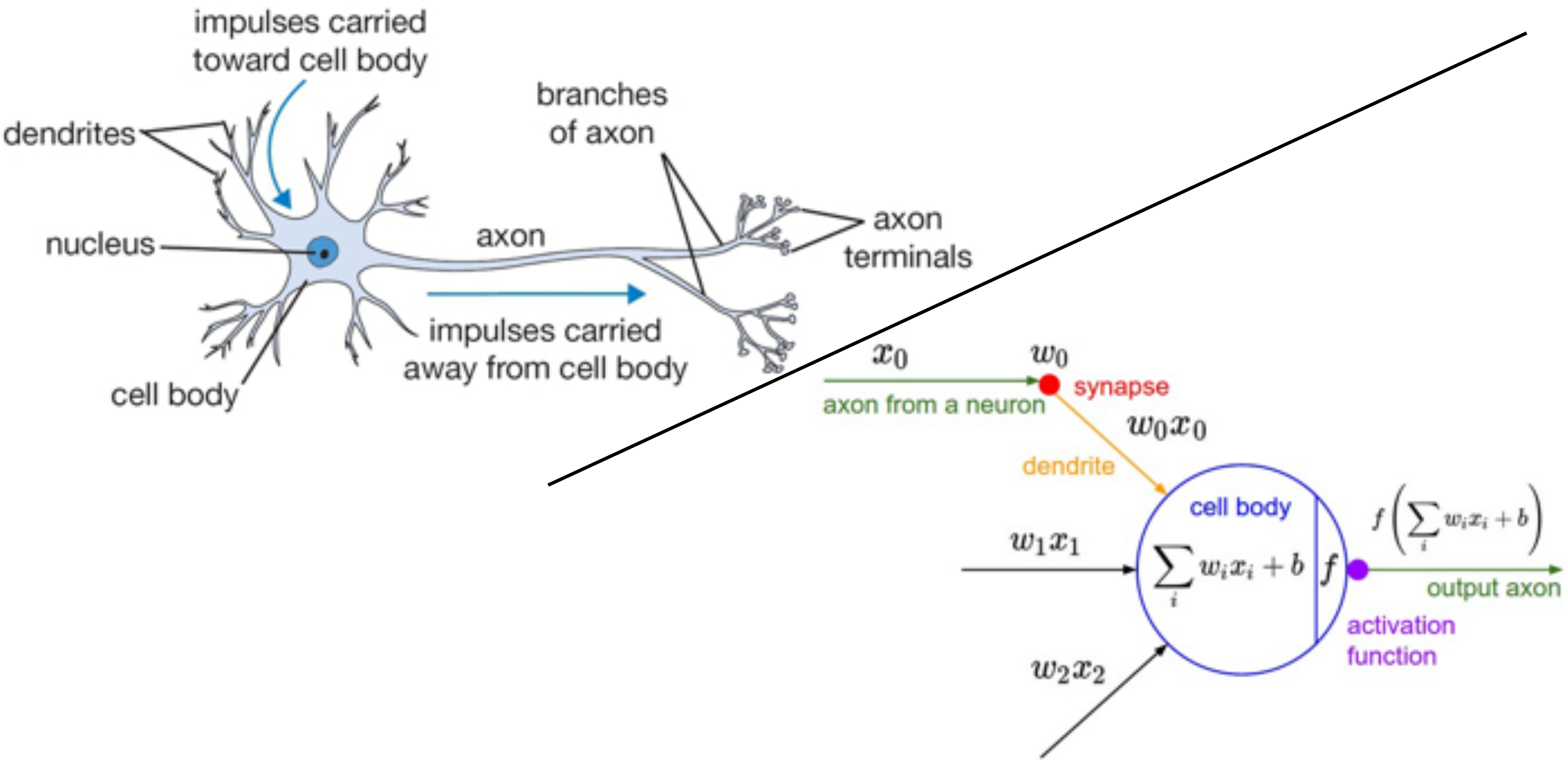
from @iamtrask, http://iamtrask.github.io/2015/07/12/basic-python-network/

Assignment: Writing 2layer Net

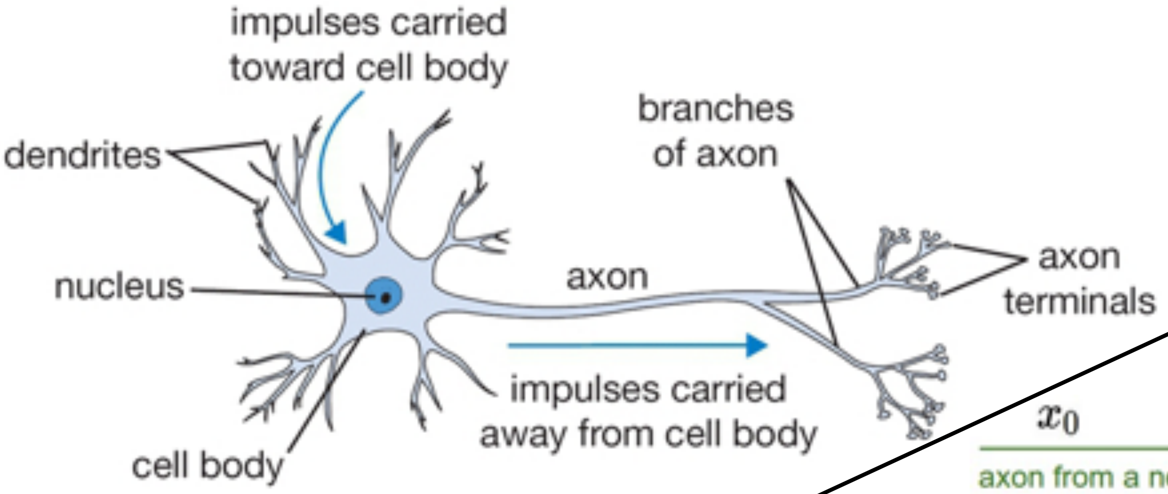
Stage your forward/backward computation!

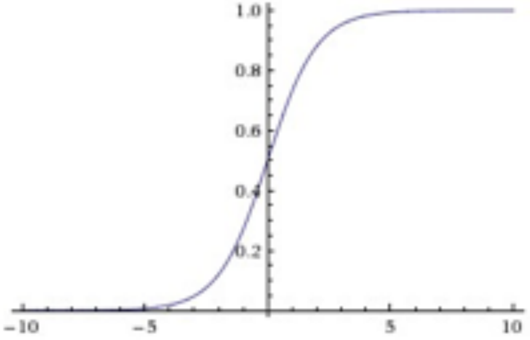
```
# receive W1,W2,b1,b2 (weights/biases), X (data)
# forward pass:
h1 = #... function of X,W1,b1
scores = #... function of h1,W2,b2
loss = #... (several lines of code to evaluate Softmax loss)
# backward pass:
dscores = #...
dh1,dW2,db2 = #...
dW1,db1 = #...
```


impulses carried
toward cell body
dendrites
branches
of axon
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body



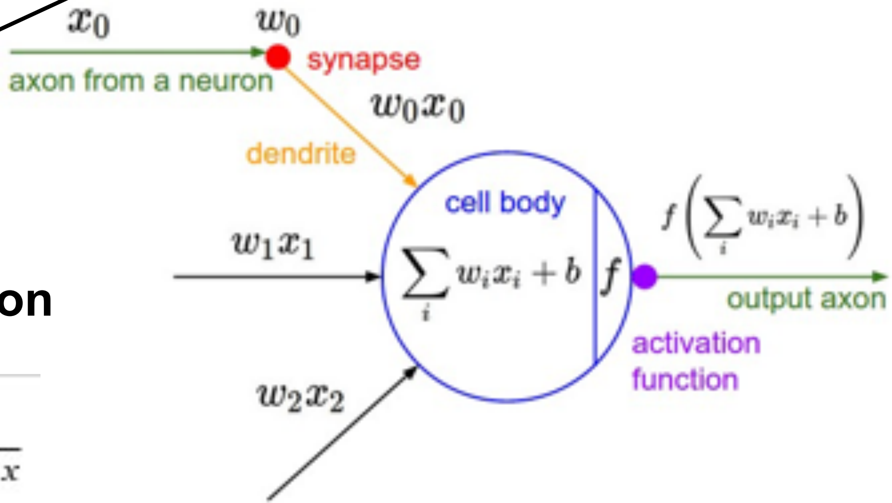
* Original slides borrowed from Andrej Karpathy
and Li Fei-Fei, Stanford cs231n

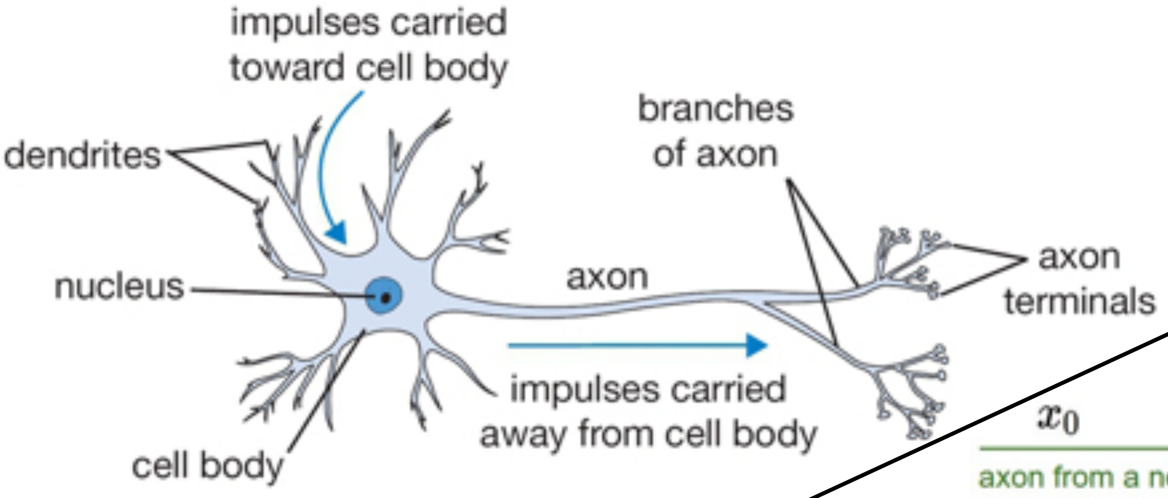


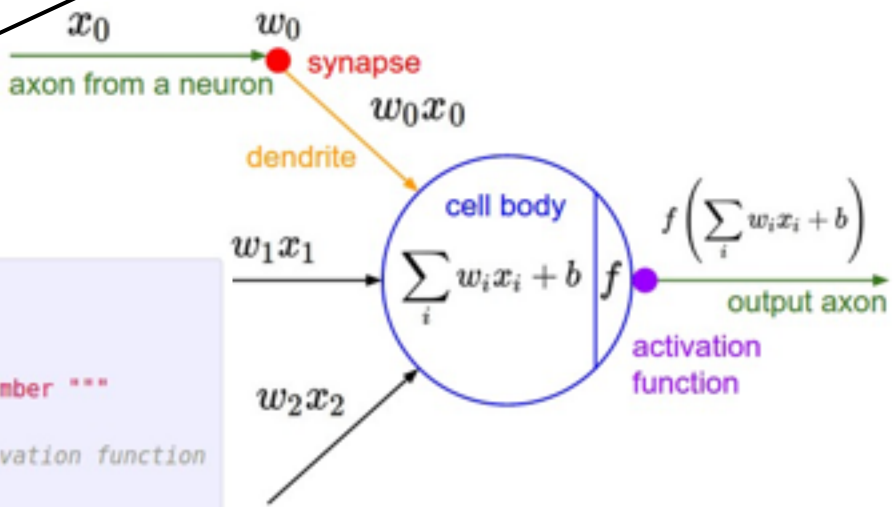


sigmoid activation function

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





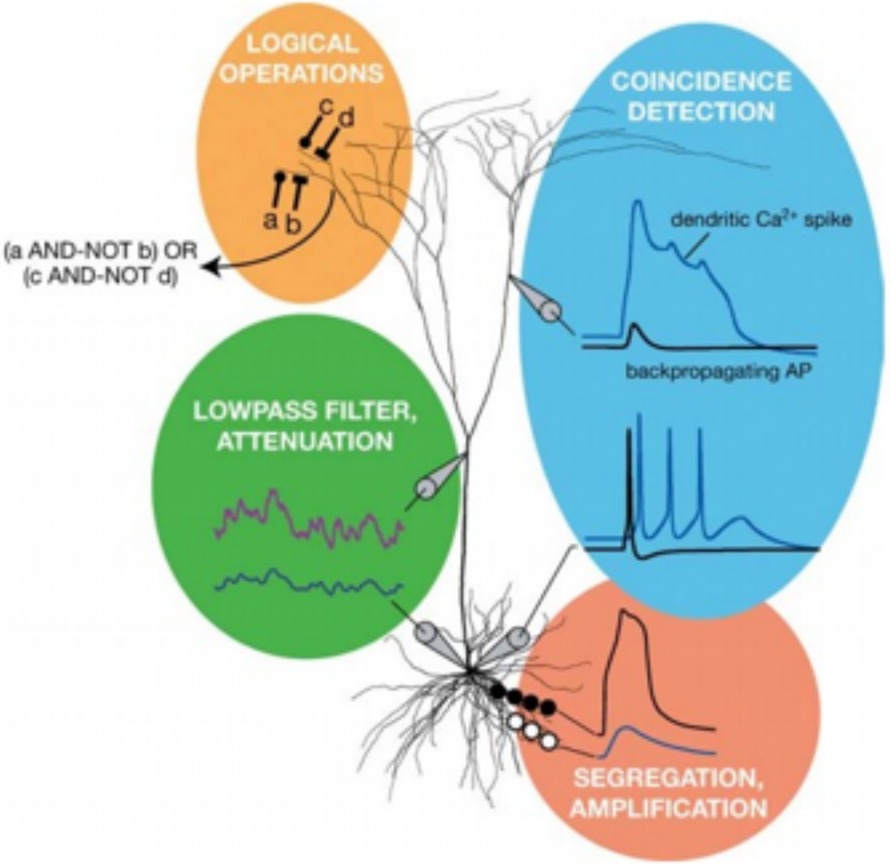



```python
class Neuron:
  # ...
  def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
    return firing_rate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

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

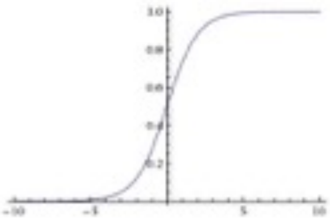


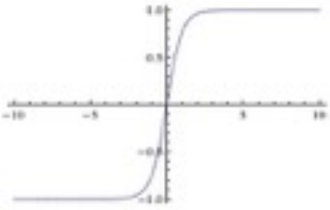
[Dendritic Computation. London and Hausser]

Activation Functions

Sigmoid

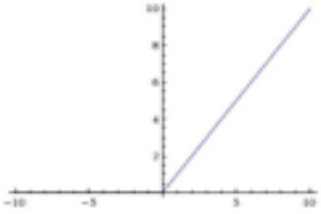
$$\sigma(x) = 1/(1+e^{-x})$$



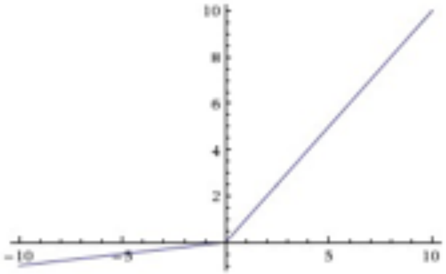


tanh tanh(x)

ReLU max(0,x)



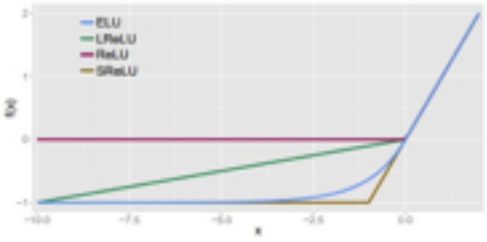
Leaky ReLU
max(0.1x, x)



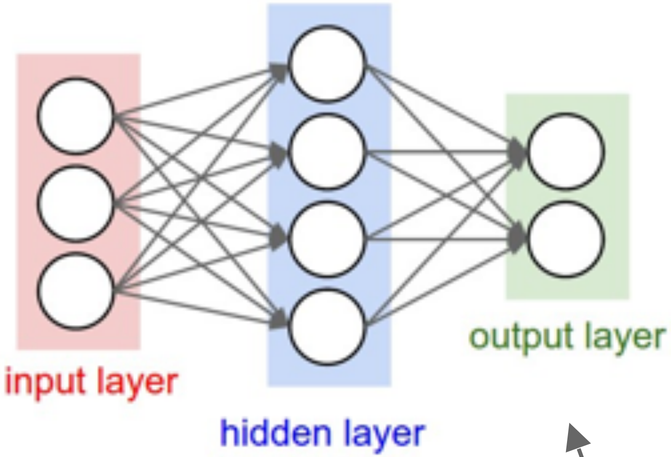
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

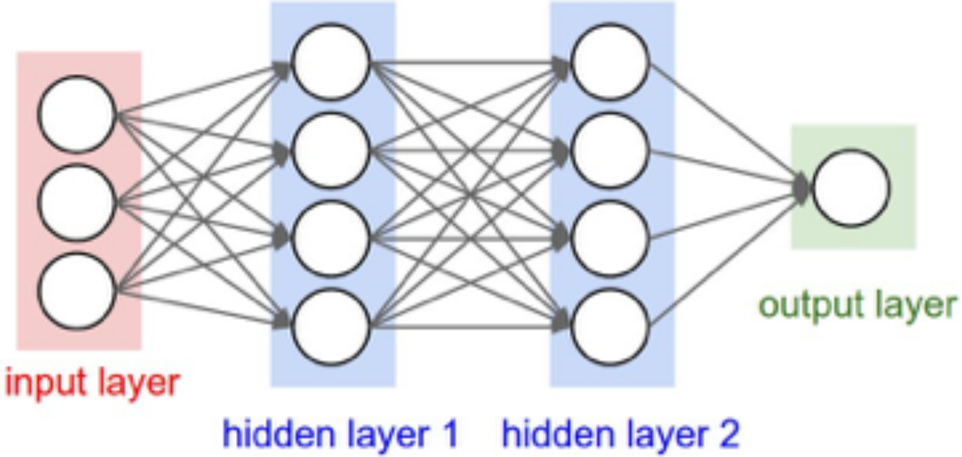
ELU

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



Neural Networks: Architectures





“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

“Fully-connected” layers

“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

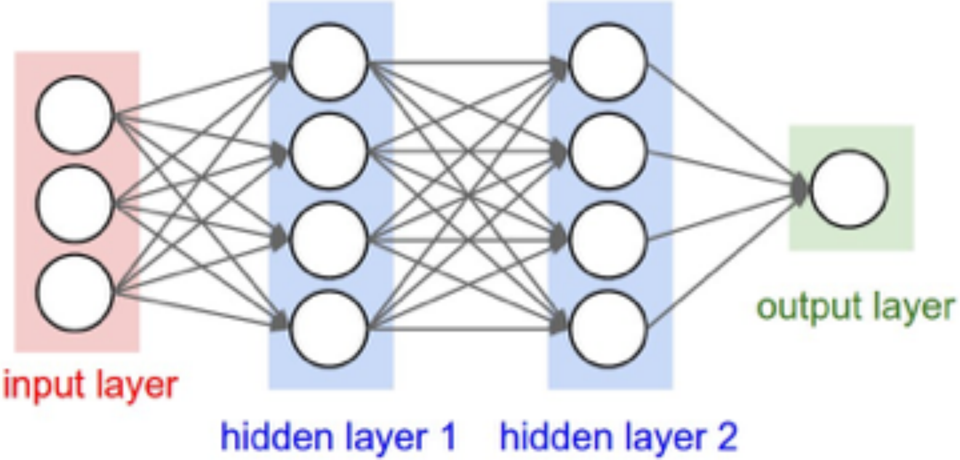
* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Example Feed-forward computation of a Neural Network

```python
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
```

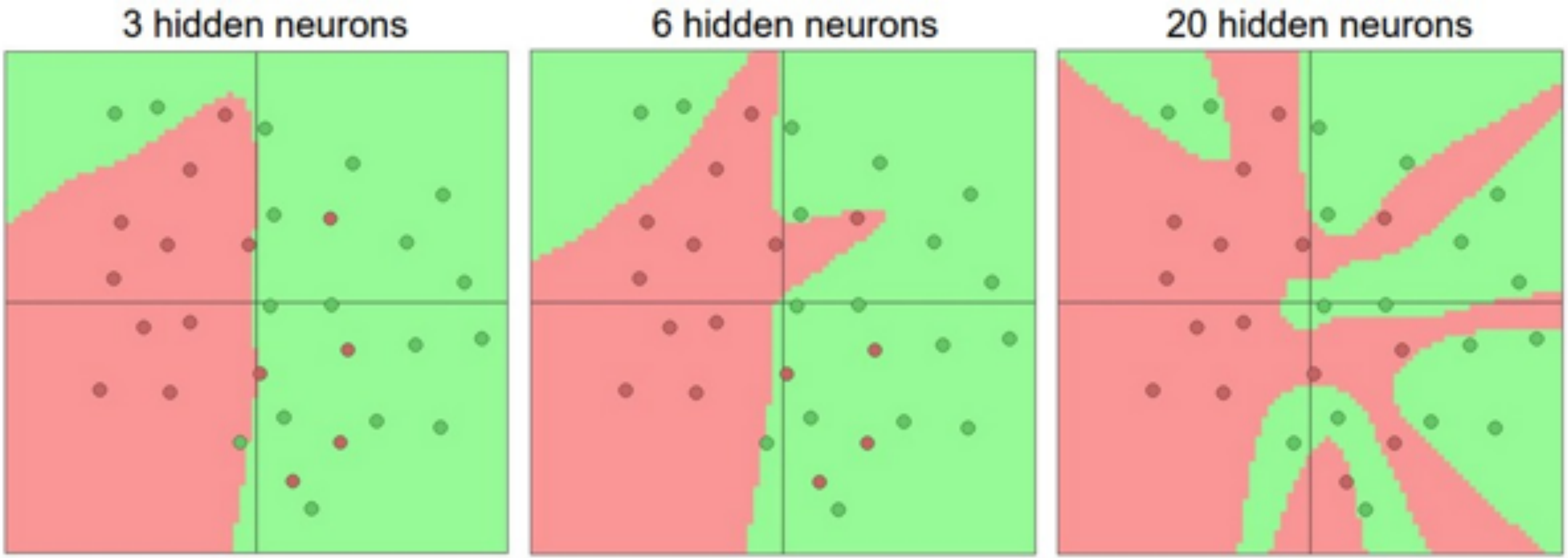
We can efficiently evaluate an entire layer of neurons.

Example Feed-forward computation of a Neural Network



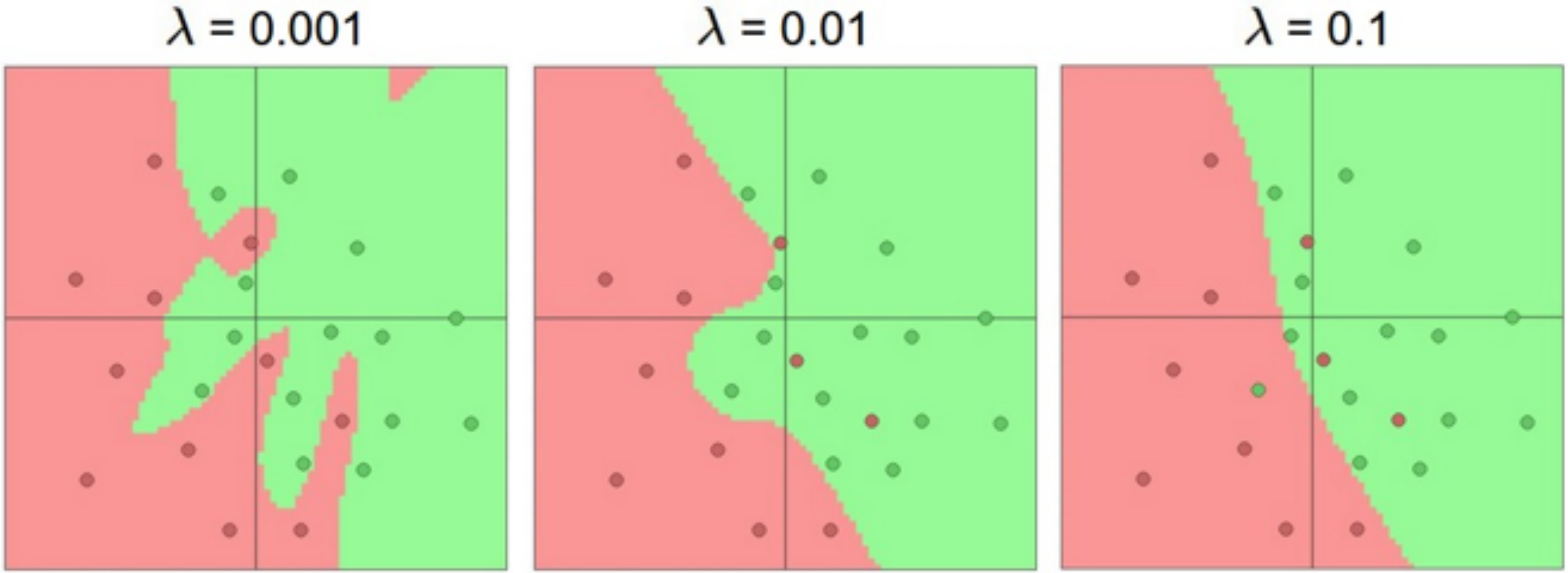

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate 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.