Backpropagation

Sudeshna Sarkar

19 Jan 2017

- Refer to notes from

- http://cs231n.github.io/optimization-2/

and slides for the corresponding course

Loss Function

- Mean square loss
- Hinge Loss
- Cross Entropy Loss

Backpropagation

- Backpropagation: a way of computing gradients of expressions through recursive application of chain rule.
- We are given some function f(x) where x is a vector of inputs and we are interested in computing the gradient of f at x (i.e. $\nabla f(x)$).
- In NN, f will correspond to the loss function (L) and the inputs x will consist of the training data and the neural network weights.
- compute the gradient for the parameters (e.g. W,b) so that we can use it to perform a parameter update.

- Scores $s = f(x; W) = Wx$
- SVM Loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$
- Data loss + regularization

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

- Want $\nabla_W L$

Optimization

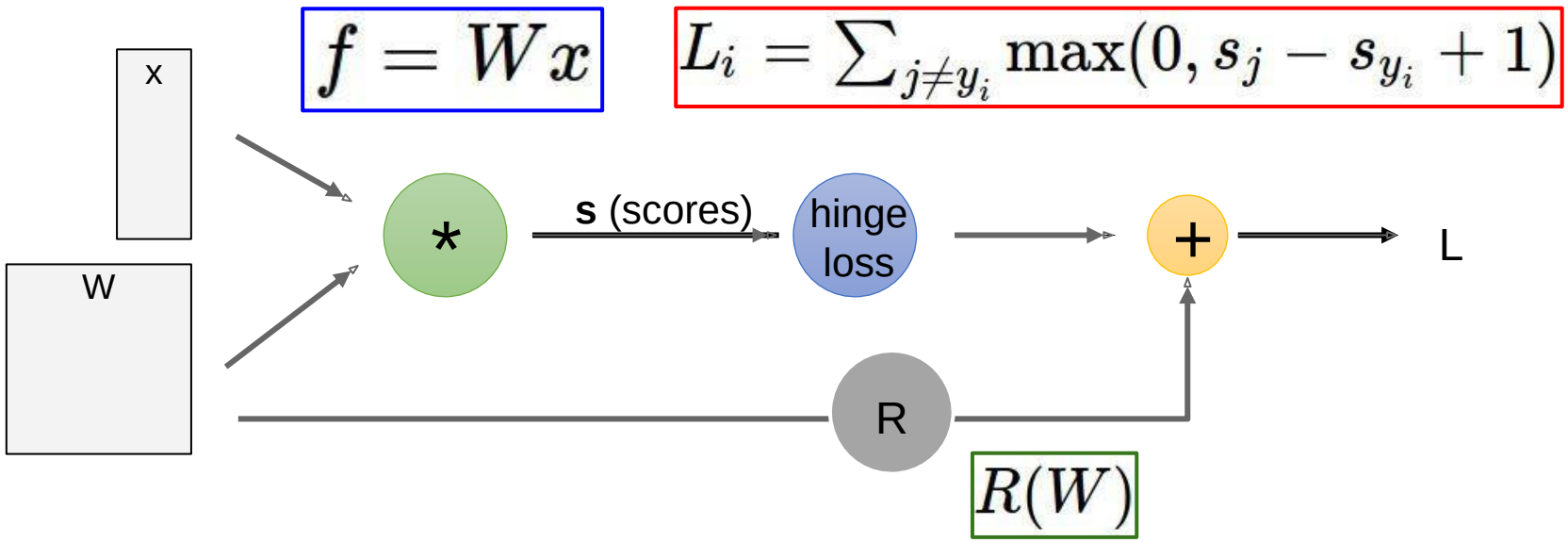
```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Gradient Descent

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

- Numerical gradient: slow, approximate, easy to write
- Analytic gradient: fast, exact, error-prone
- In practice: Derive analytic gradient, check your implementation with numerical gradient

Computational Graph

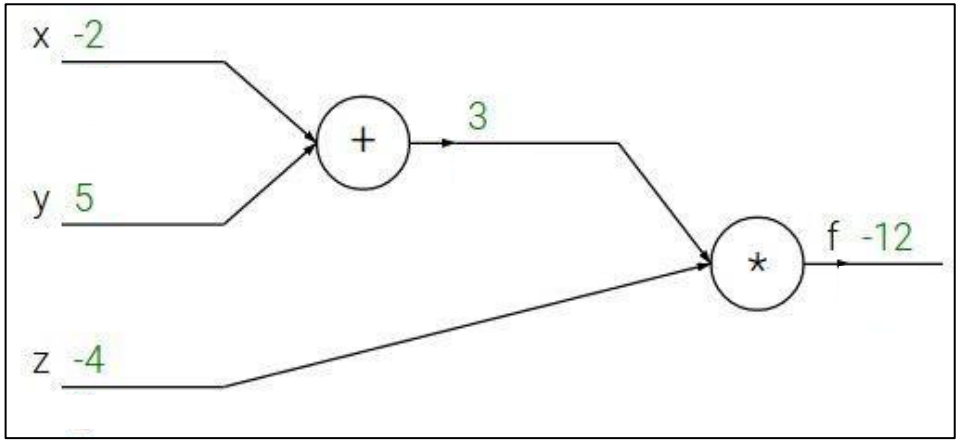


Intuitive understanding of backpropagation

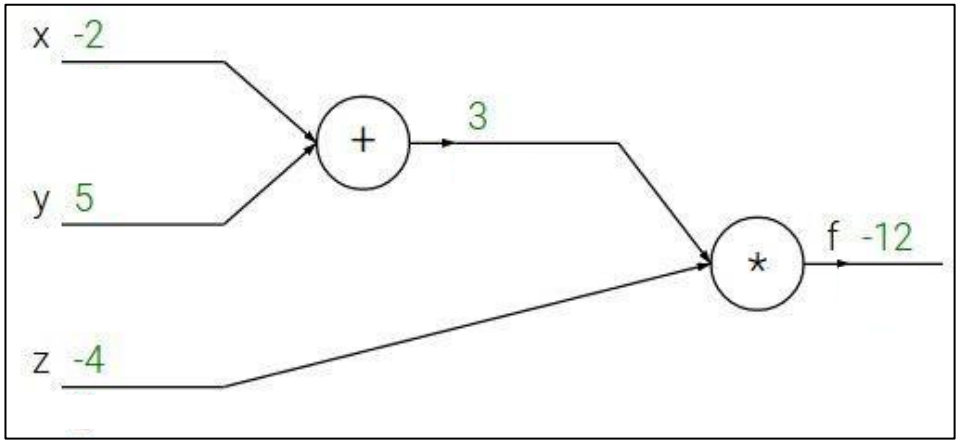
- Backpropagation: local process.
- Every gate gets some inputs and can compute two things:
  1. its output value
  2. the local gradient of its inputs with respect to its output value.
- once the forward pass is over, during backpropagation the gate will eventually learn about the gradient of its output value on the final output of the entire circuit.
- Chain rule : the gate should take that gradient and multiply it into every gradient it normally computes for all of its inputs.

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

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



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



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

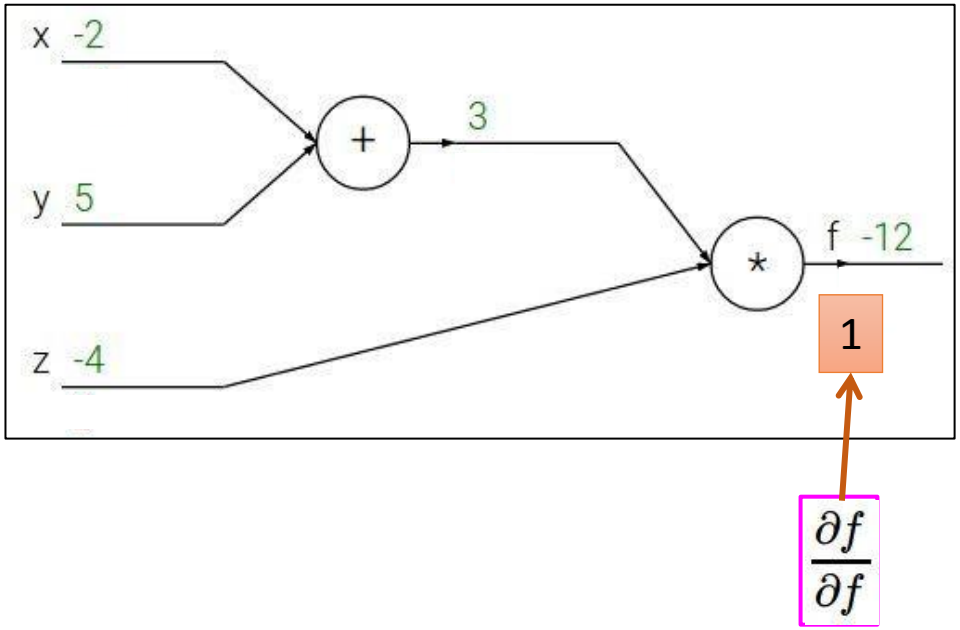
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

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

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



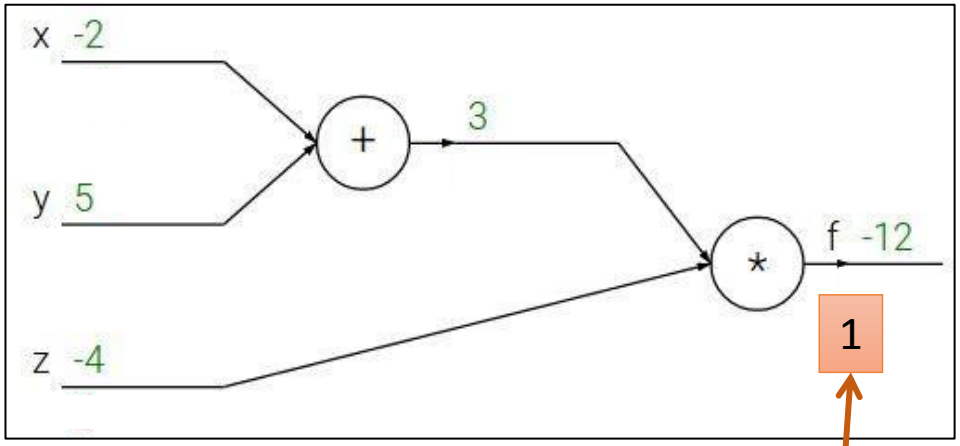
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

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

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

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

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

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

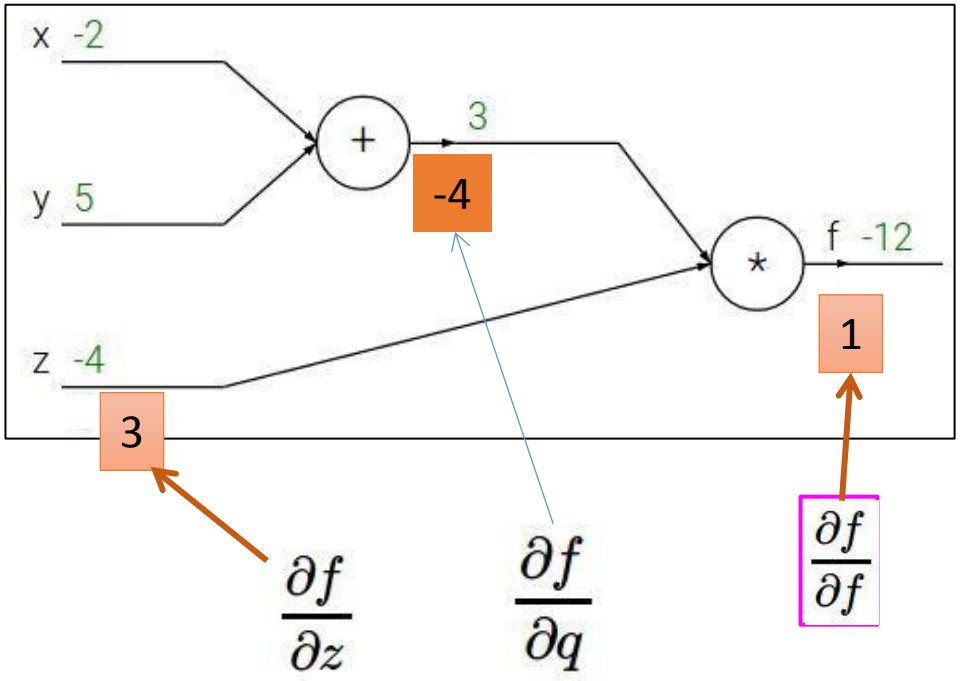
$$f(x, y, z) = (x + y)z$$

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

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

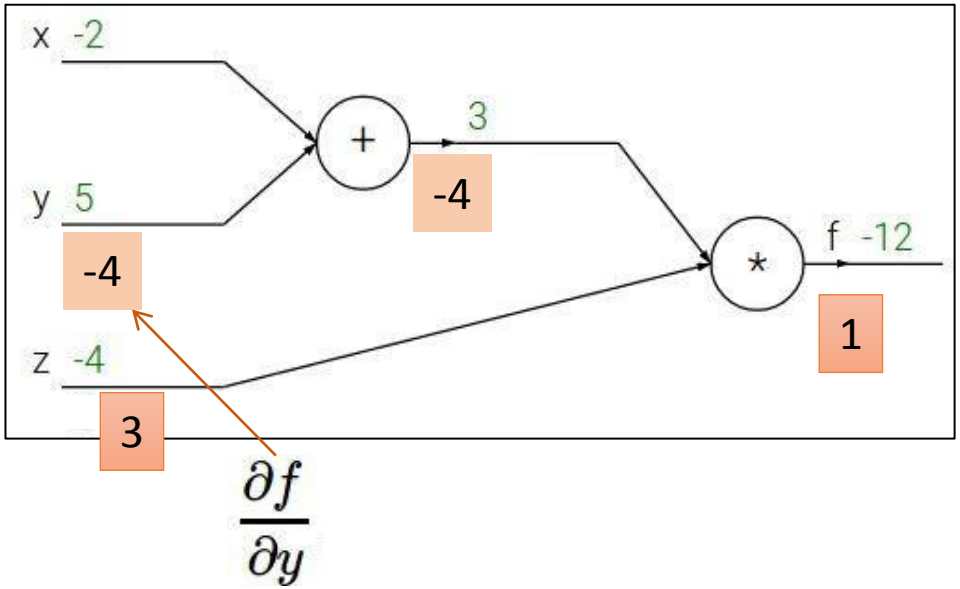
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

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



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

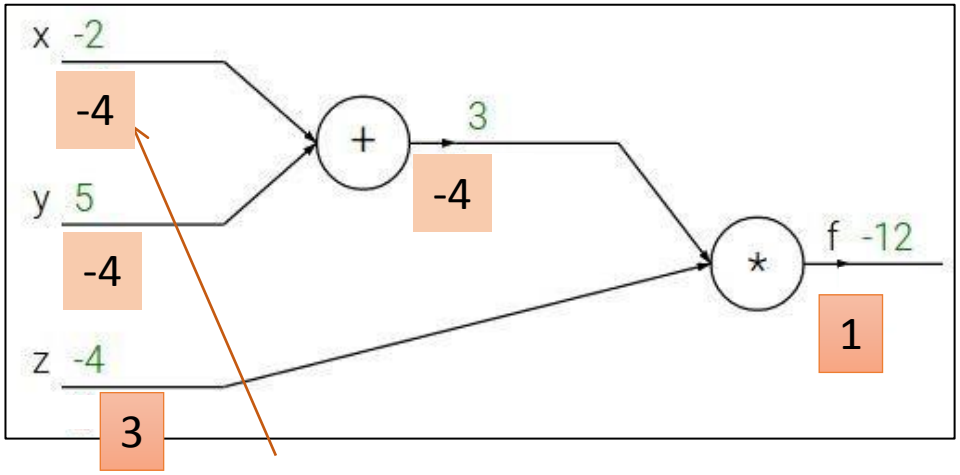
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain Rule: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$

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



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

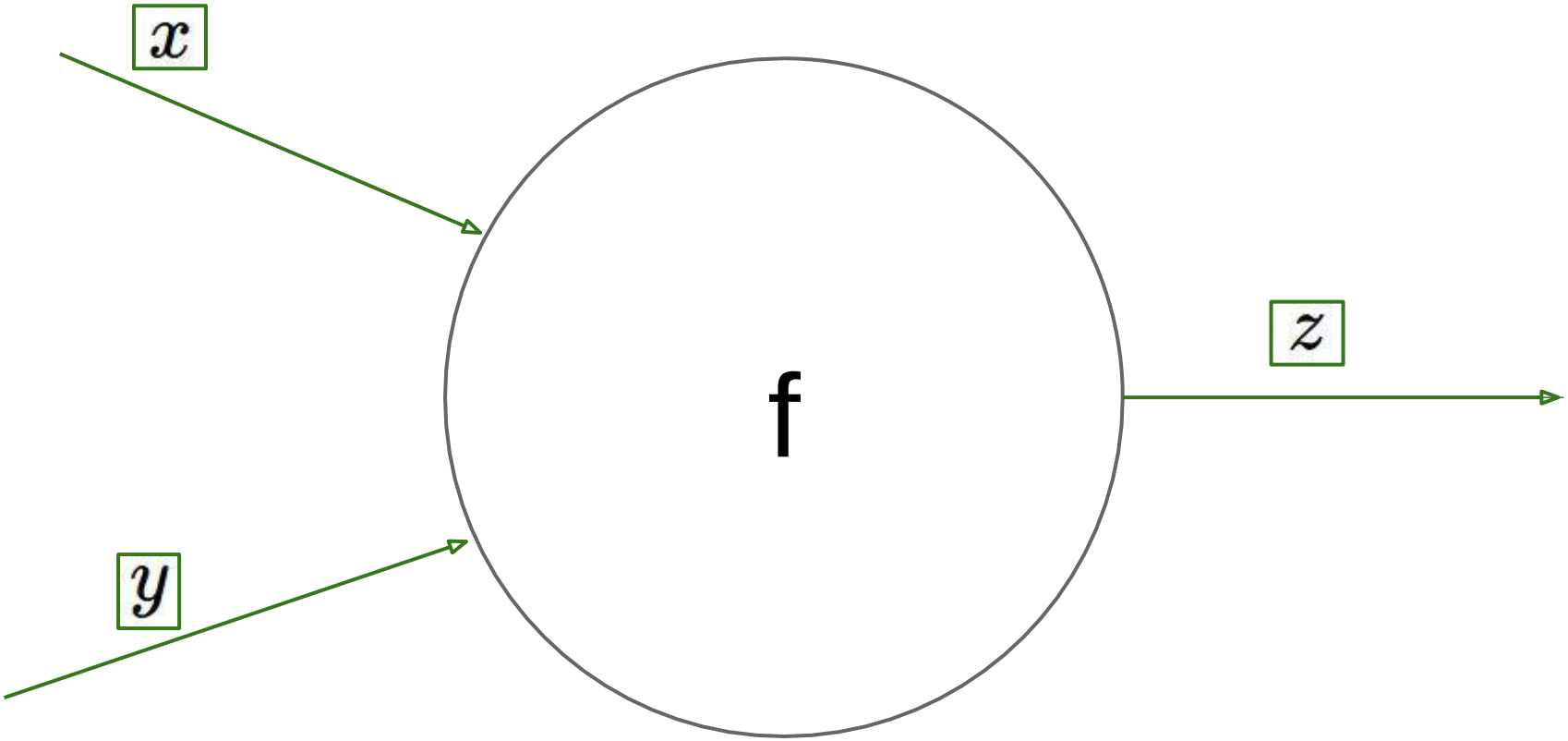
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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

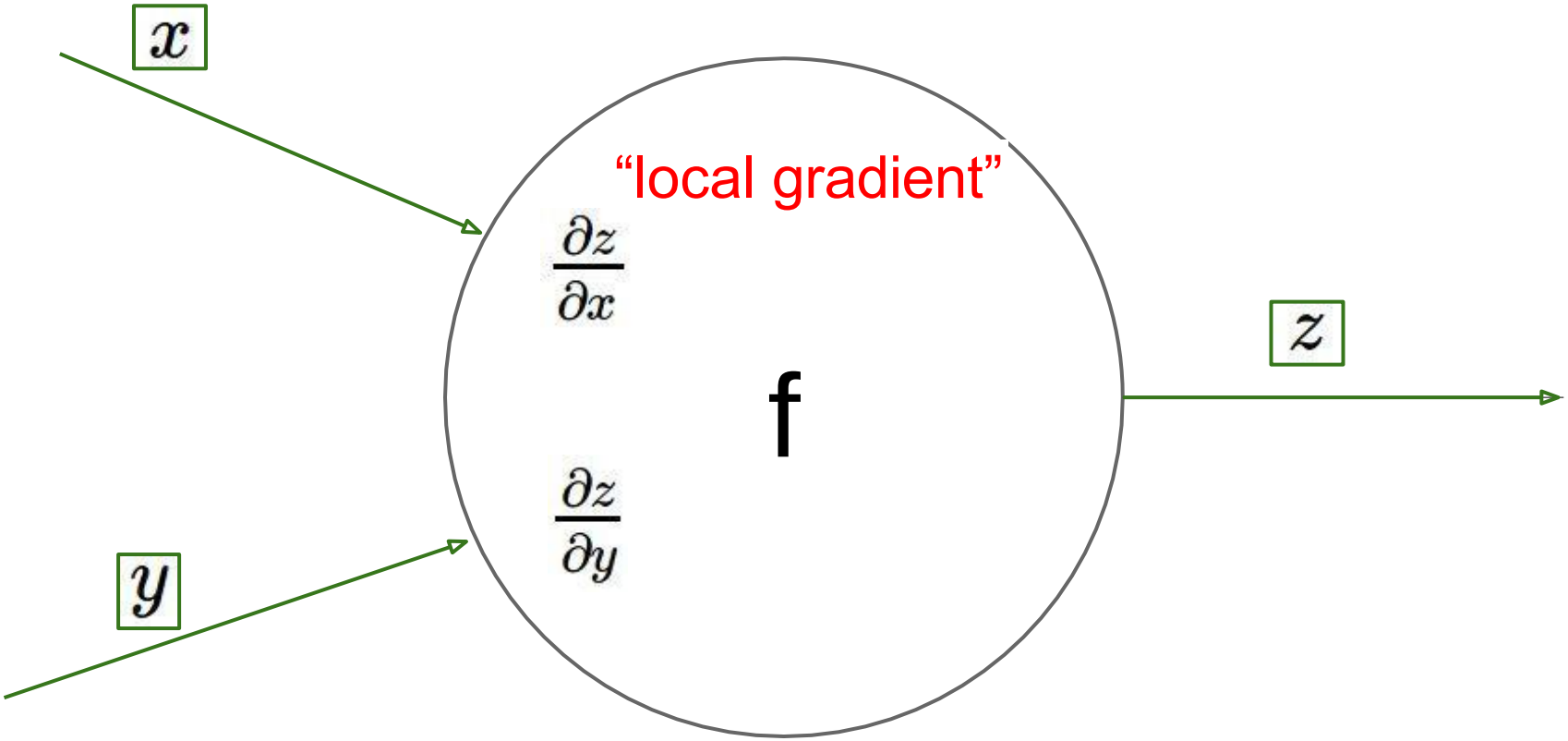
Chain Rule: $$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

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

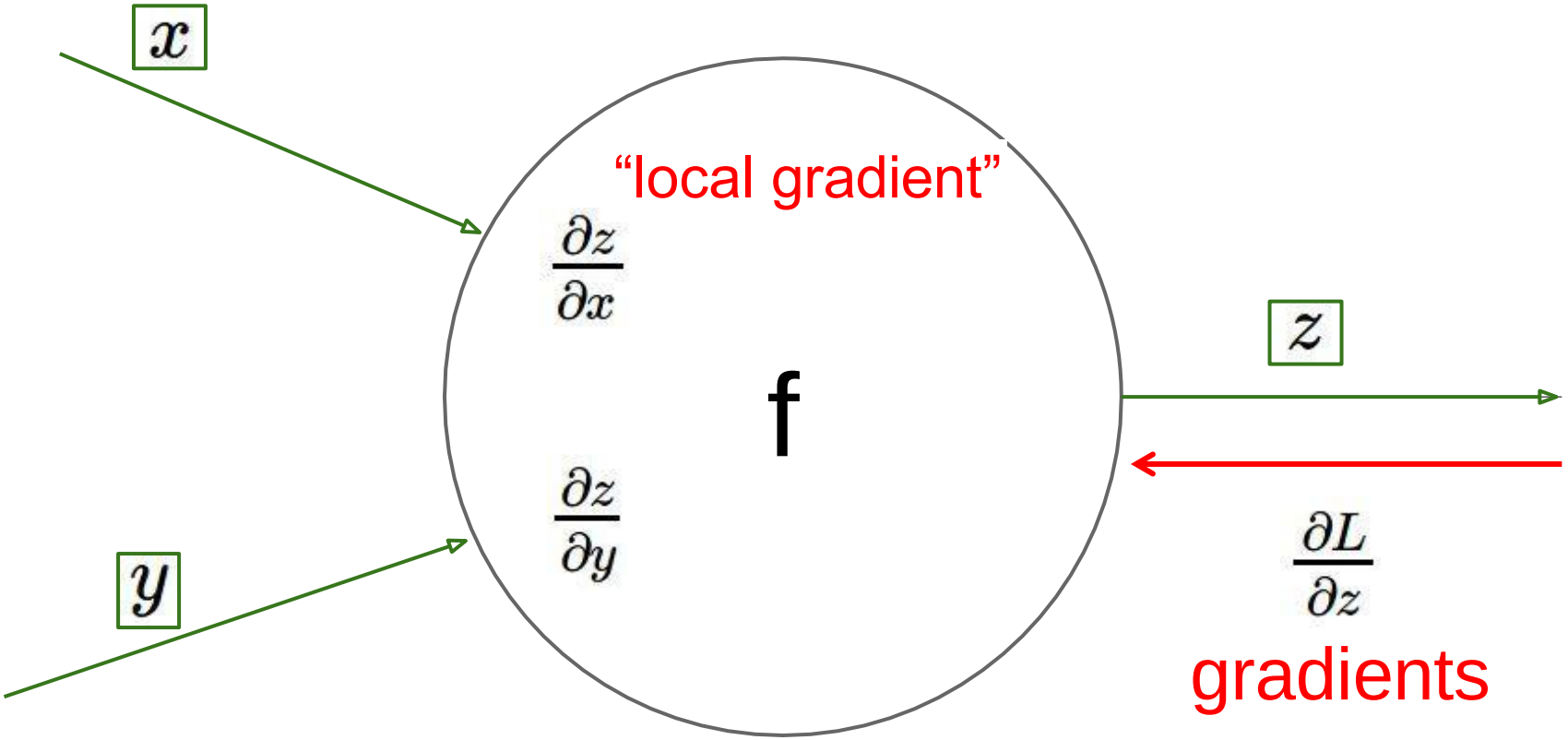
Activations



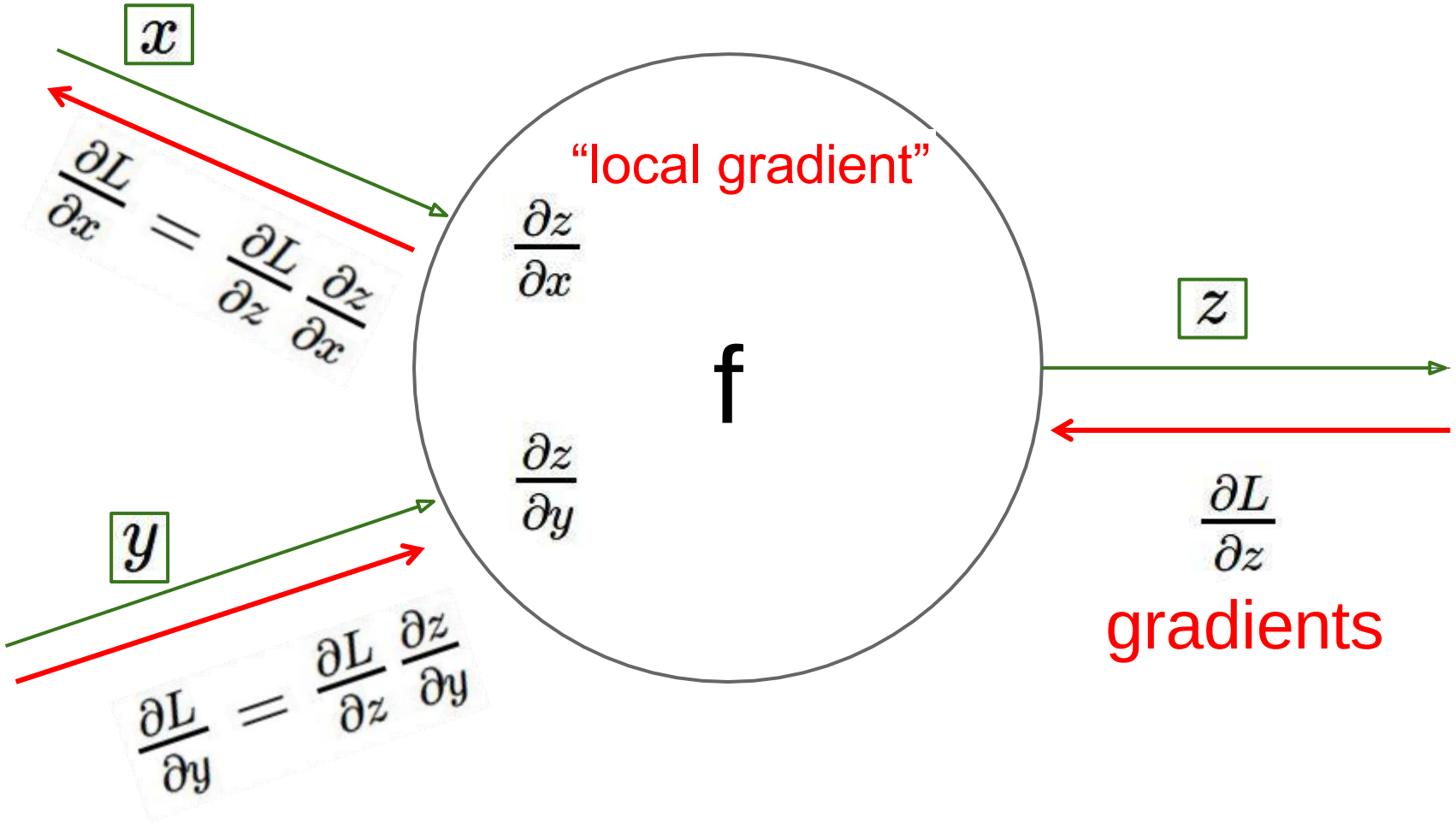
Activations



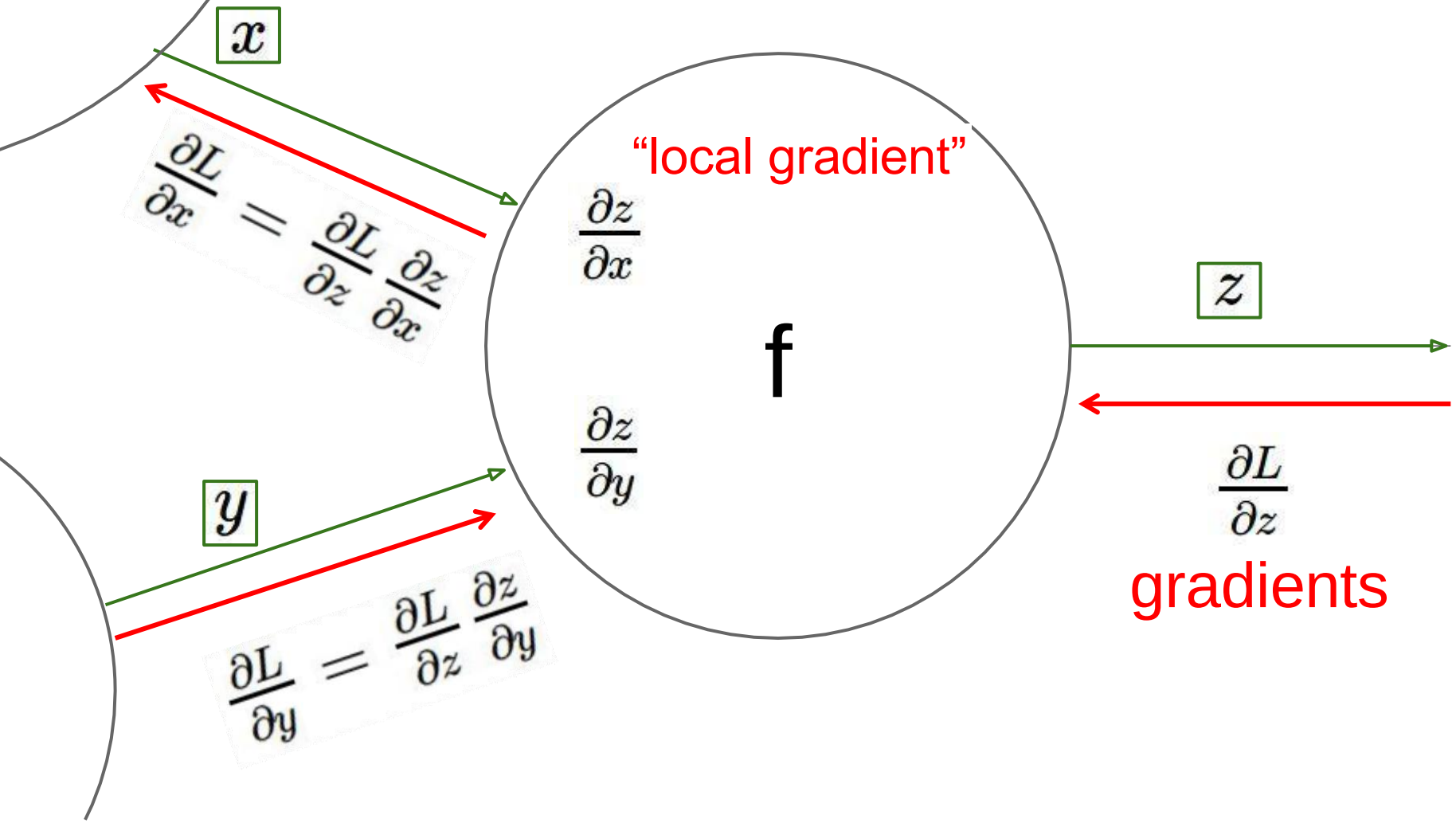
Activations



Activations

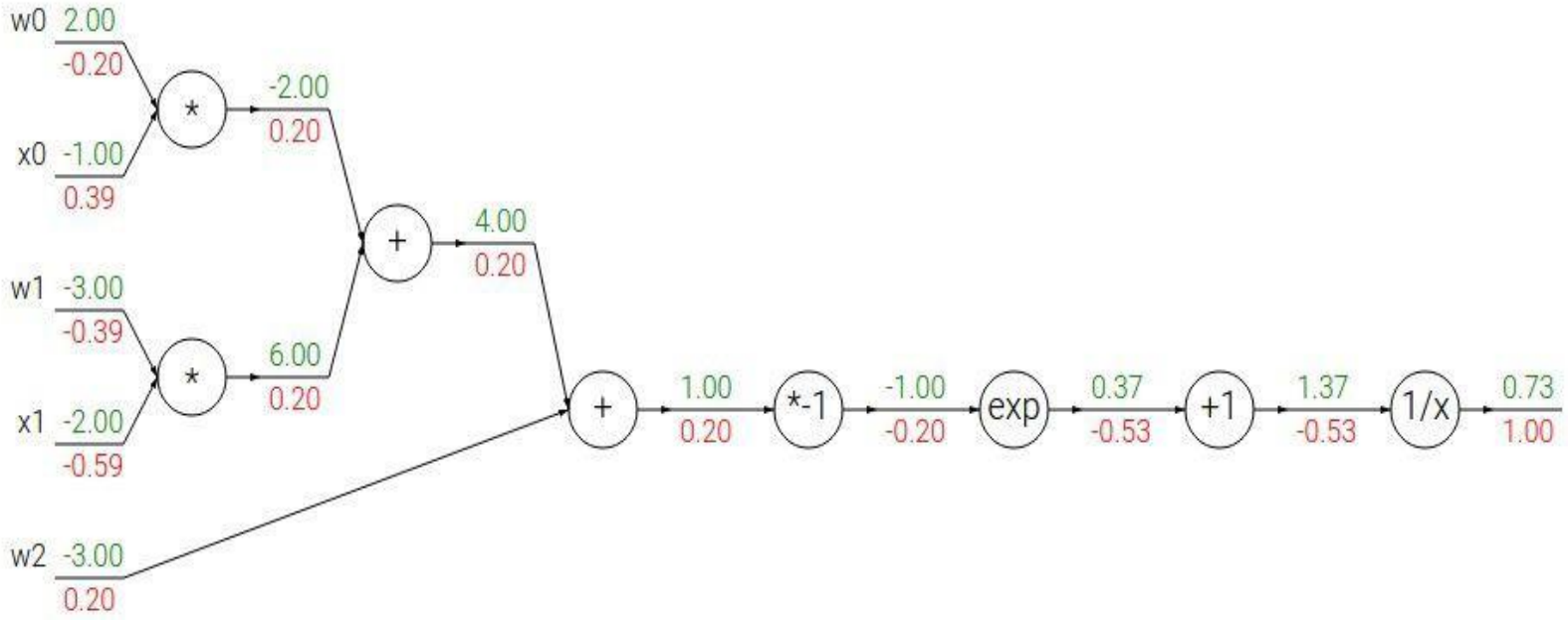


Activations



Another example

$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



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

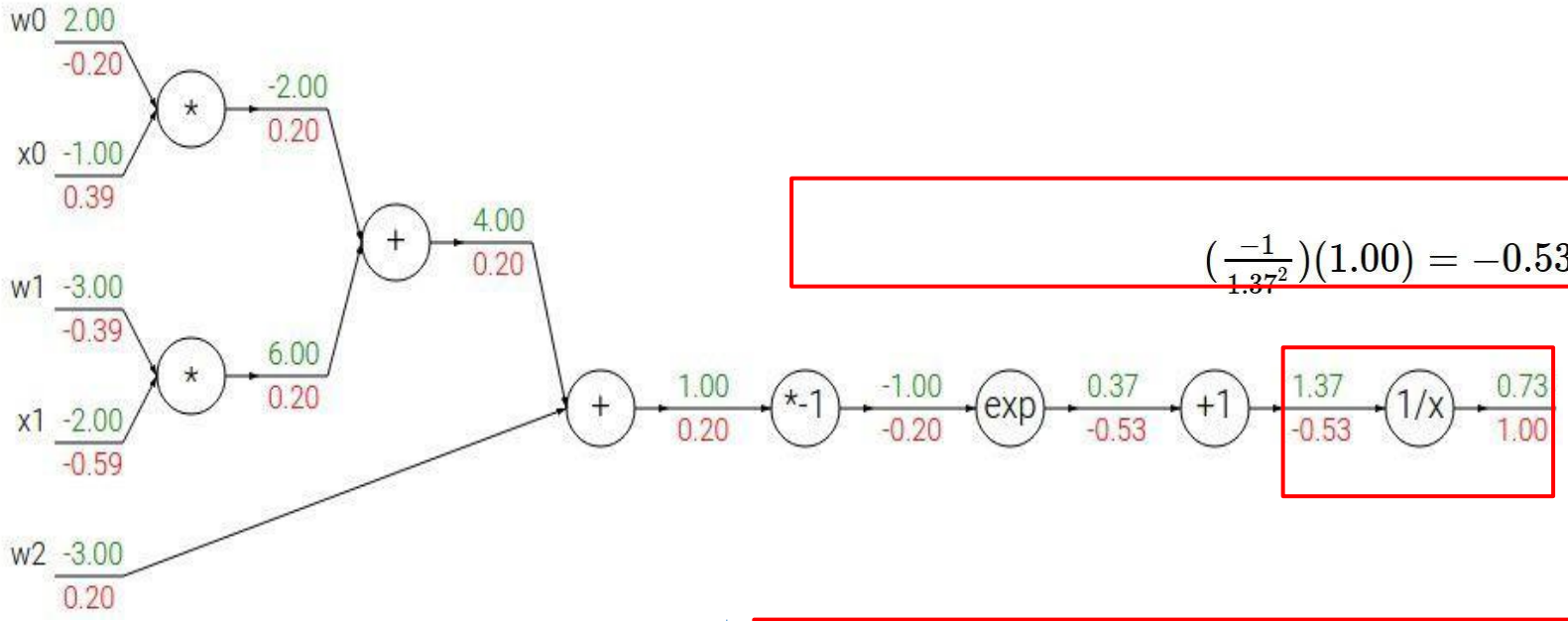
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example

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



$$(\tfrac{-1}{1.37^2})(1.00) = -0.53$$

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

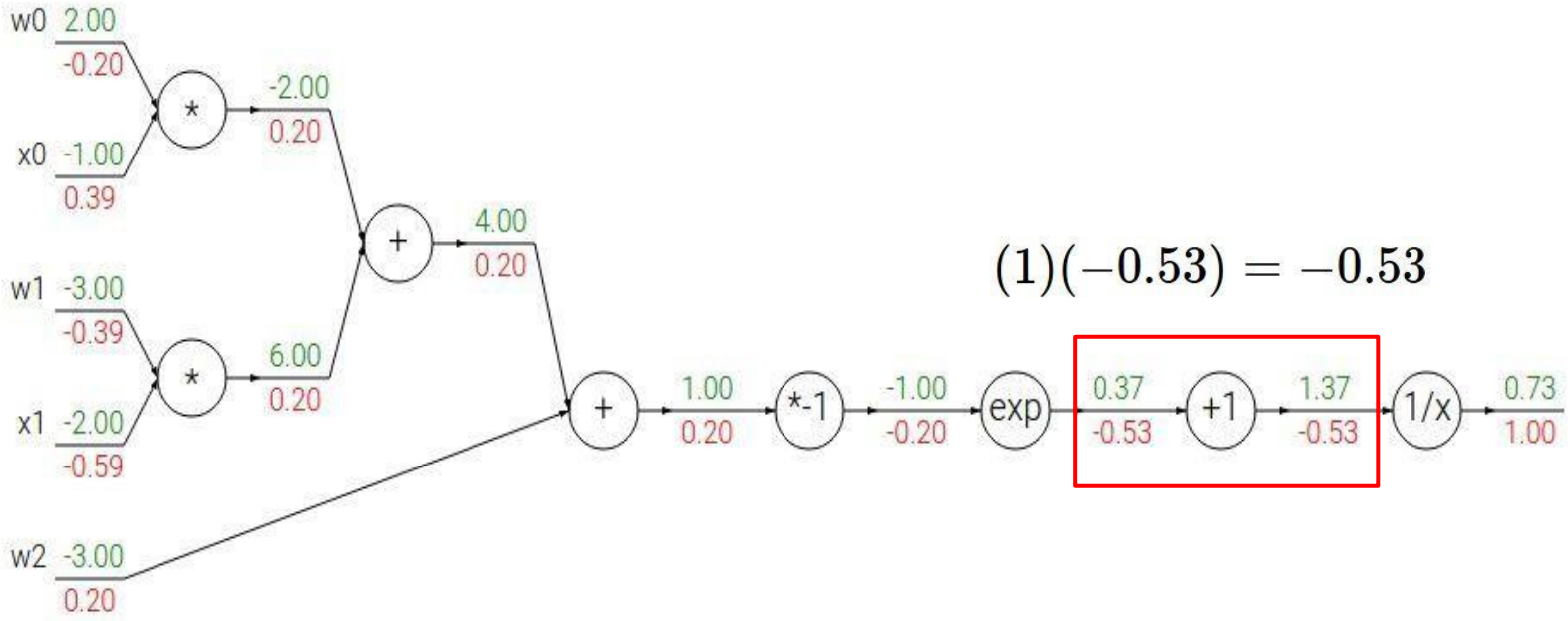
$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

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

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

Another example

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



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

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

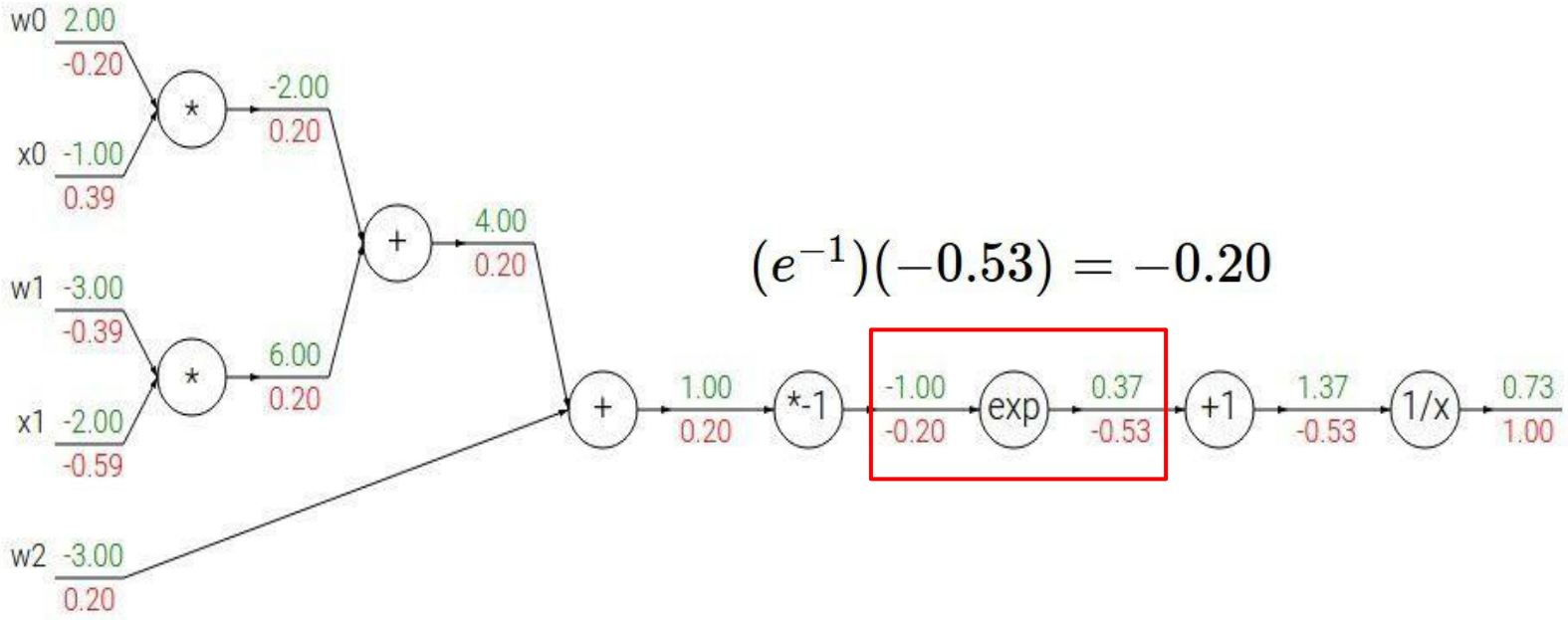
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example

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



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

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

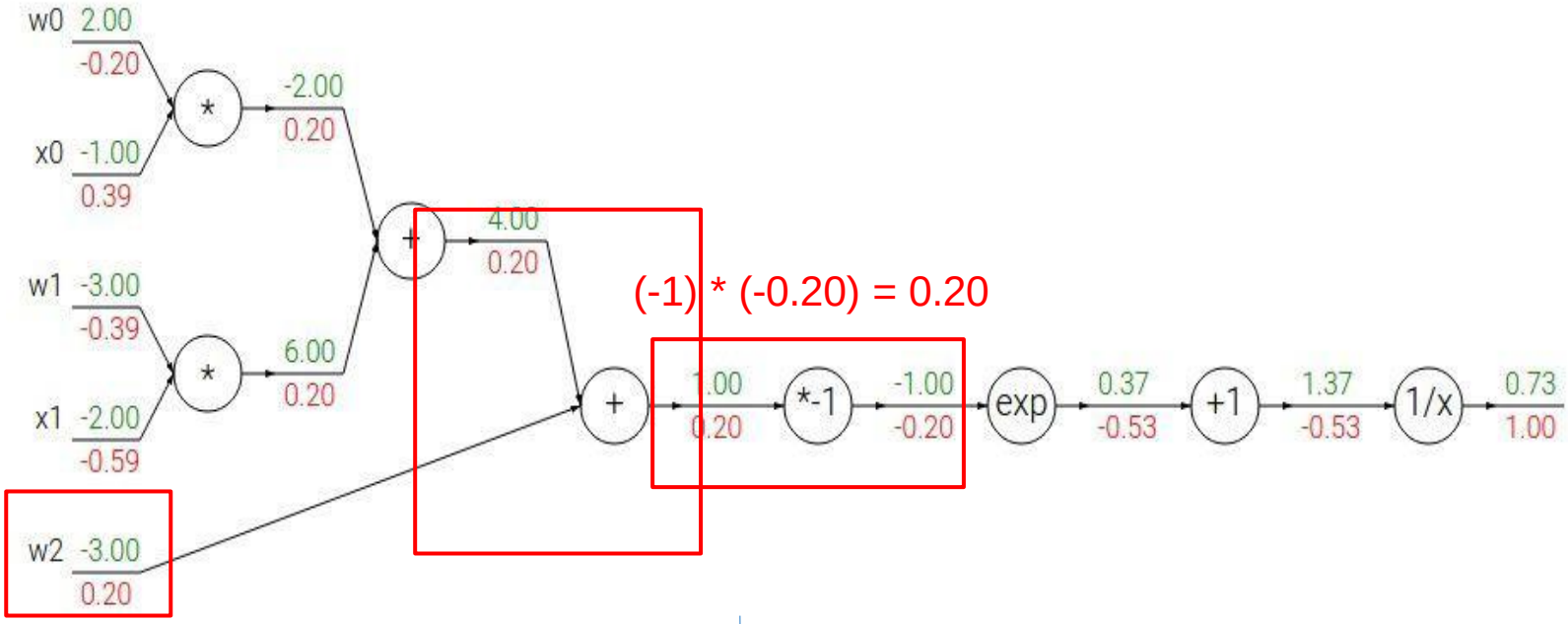
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example

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



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

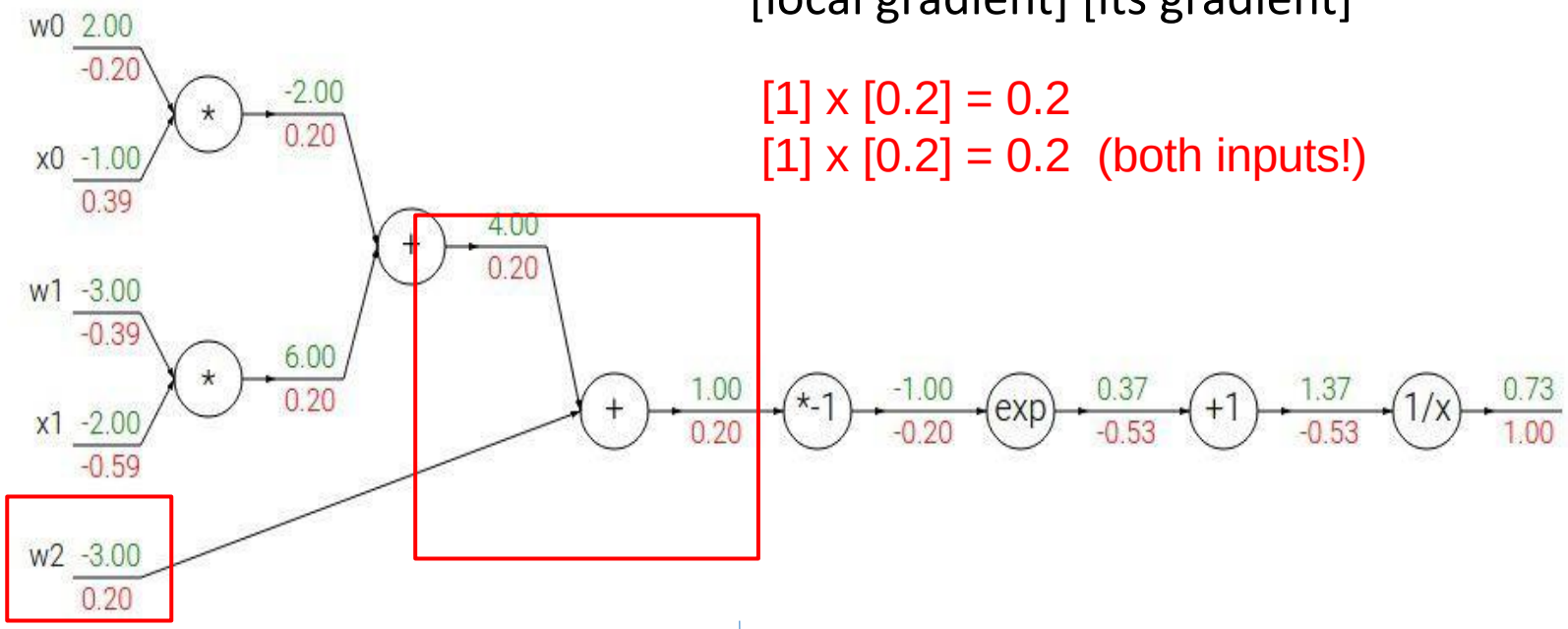
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

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

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

Another example

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



[local gradient] [its gradient]

[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)

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

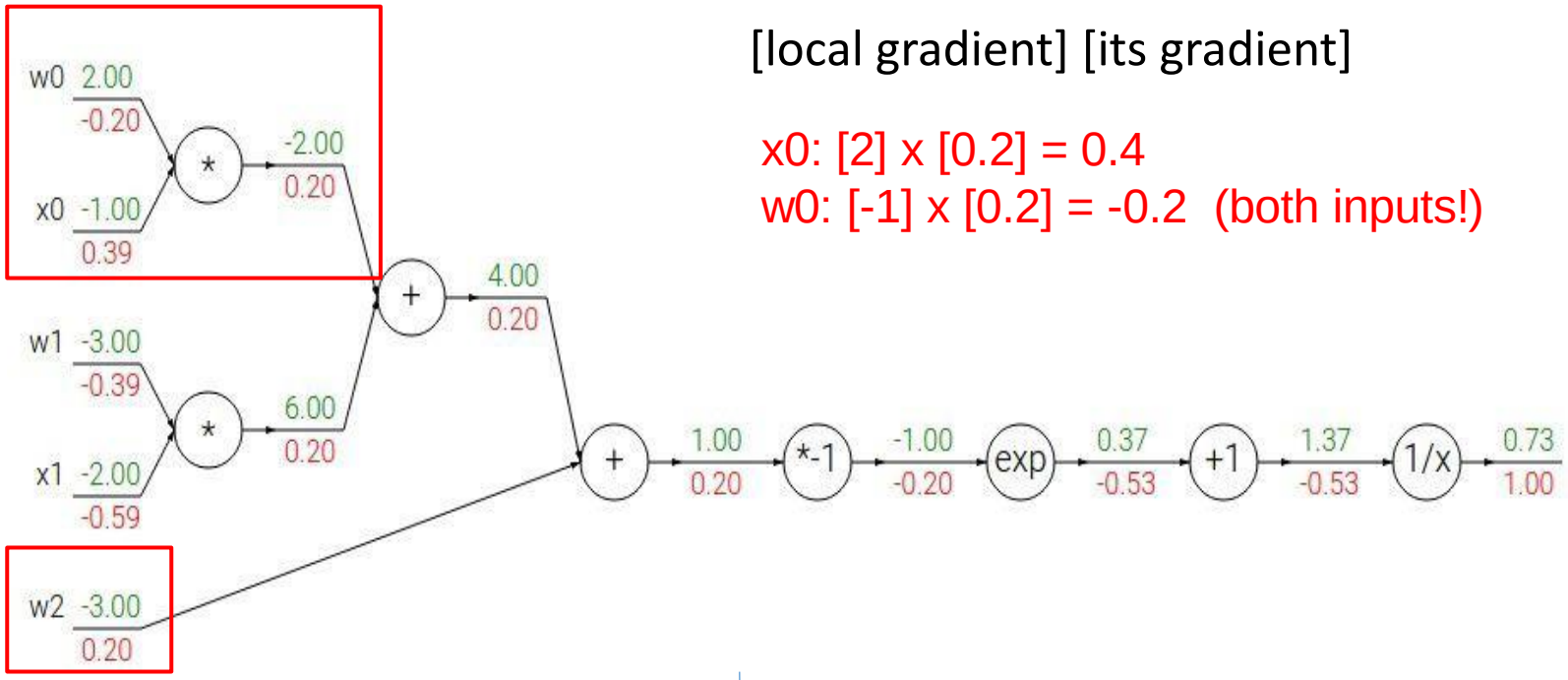
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

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

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

Another example

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



[local gradient] [its gradient]

x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2 (both inputs!)

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

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

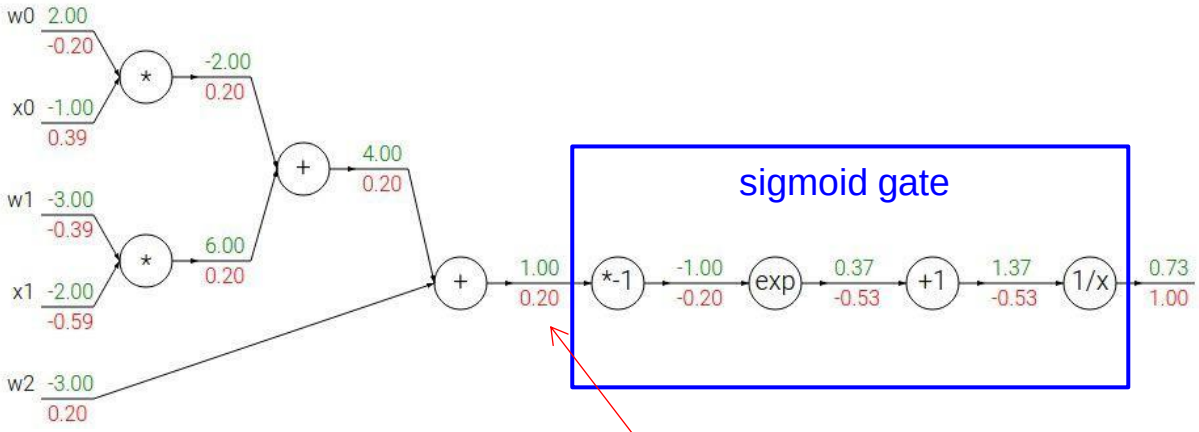
$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

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

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

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

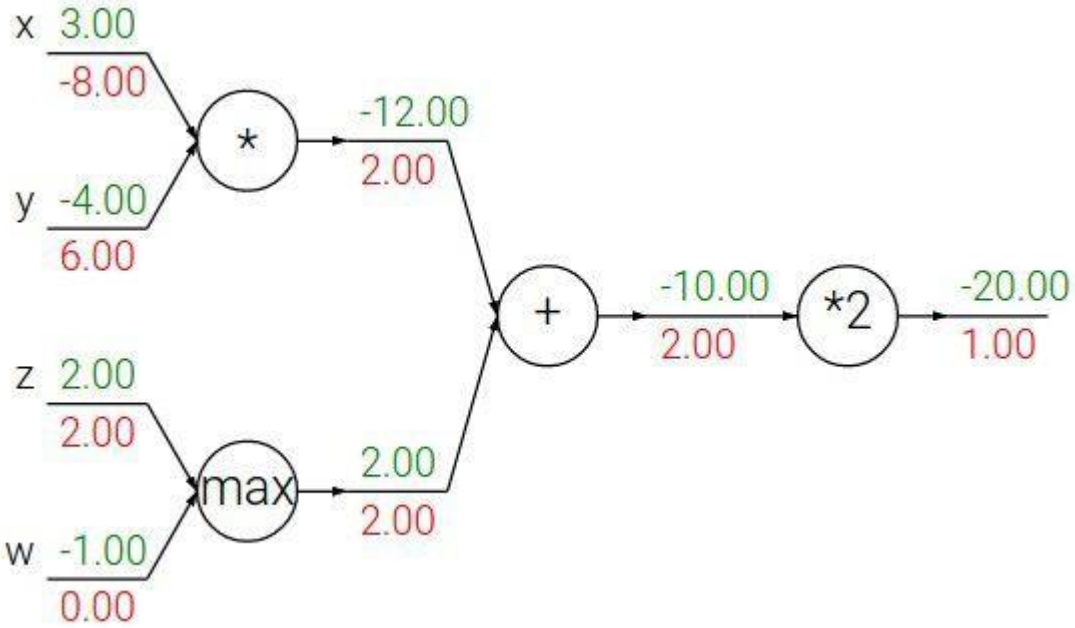
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$



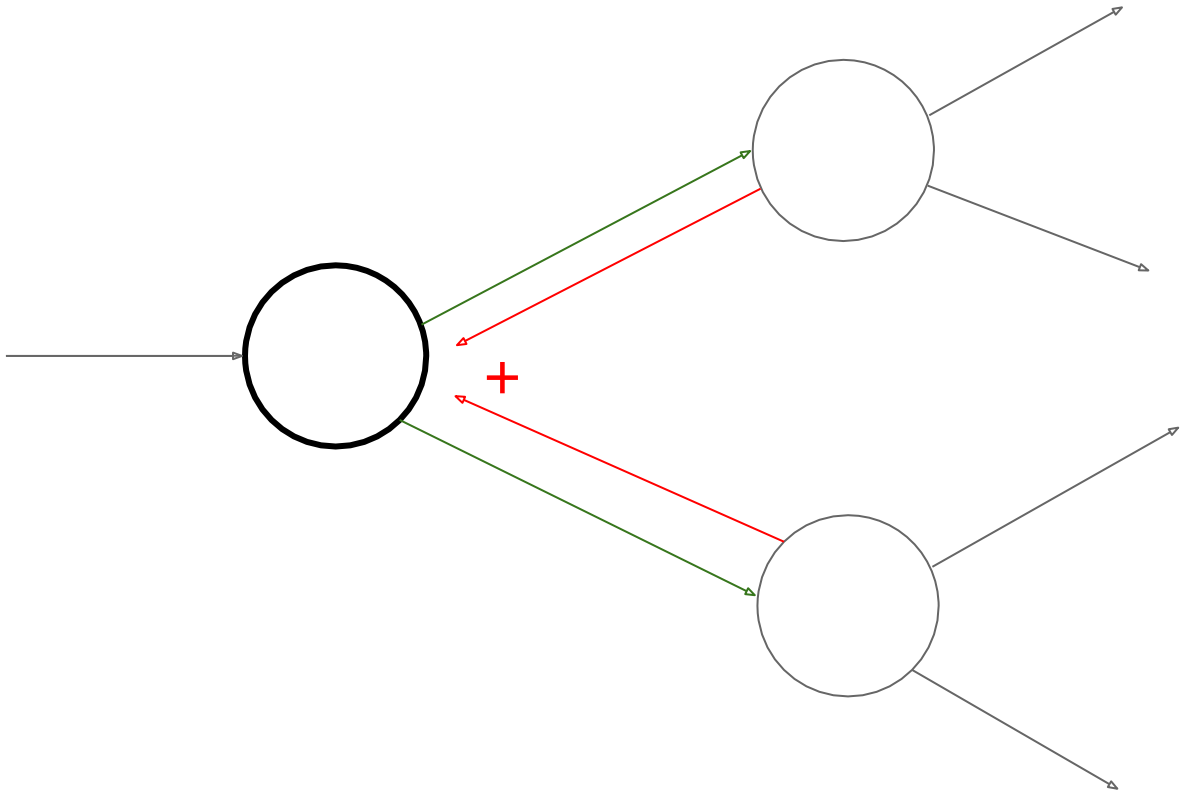
(0.73) * (1 - 0.73) = 0.2

Patterns in backward flow

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

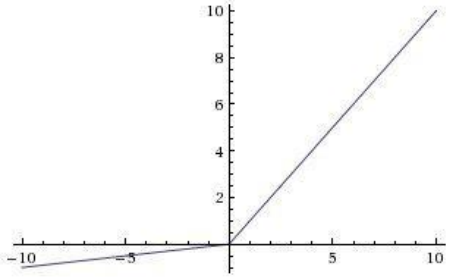


Gradients add at branches

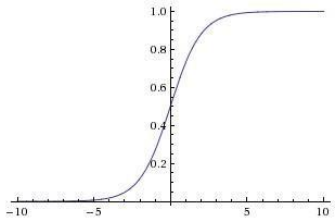


Activation functions

Leaky RELU
max(0.1x, x)



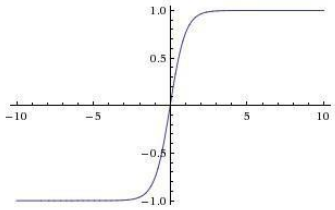
$$\sigma(x) = 1/(1 + e^{-x})$$



Maxout:

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

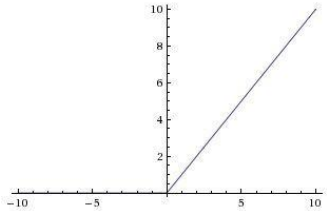
tanh: tanh(x)

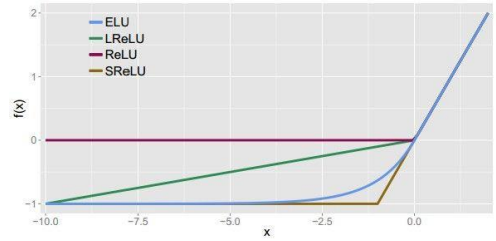


ELU:

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

ReLU: max(0,x)





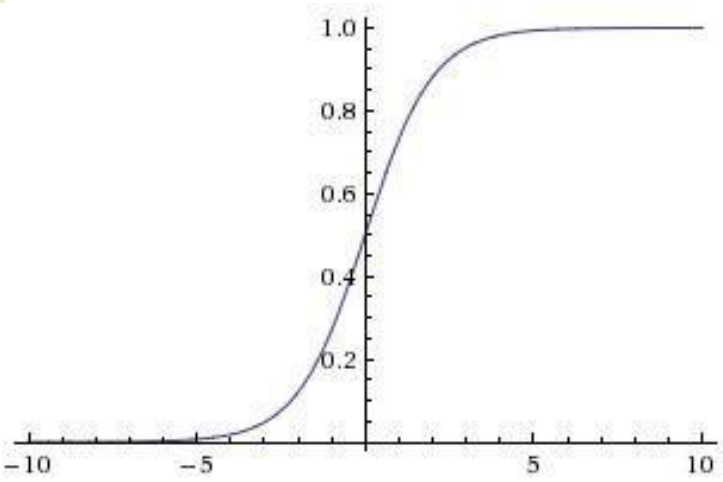
Training Neural Networks

Activation Function : Sigmoid

Squashes numbers to range [0,1]

- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

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



3 problems:

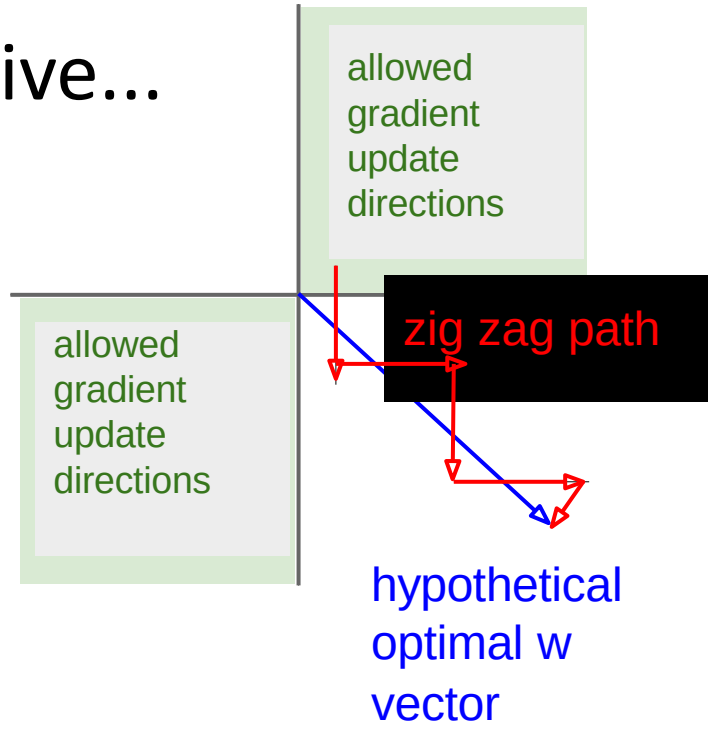
1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are non zero-centred
3. exp() is a bit compute expensive

Sigmoids saturate and kill gradients.

- when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero.
- During backpropagation, this (local) gradient will be multiplied to the gradient of this gate's output for the whole objective.
- Therefore, if the local gradient is very small, it will effectively "kill" the gradient and almost no signal will flow through the neuron to its weights and recursively to its data.
- Additionally, one must pay extra caution when initializing the weights of sigmoid neurons to prevent saturation. For example, if the initial weights are too large then most neurons would become saturated and the network will barely learn.

Consider what happens when the input to a neuron is always positive...

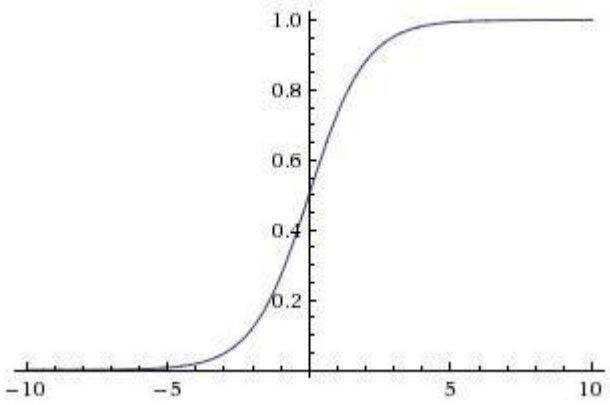
$$f\left(\sum_i w_i x_i + b\right)$$



Always all positive or all negative
(this is also why you want zero-mean data!)

Activation function tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated

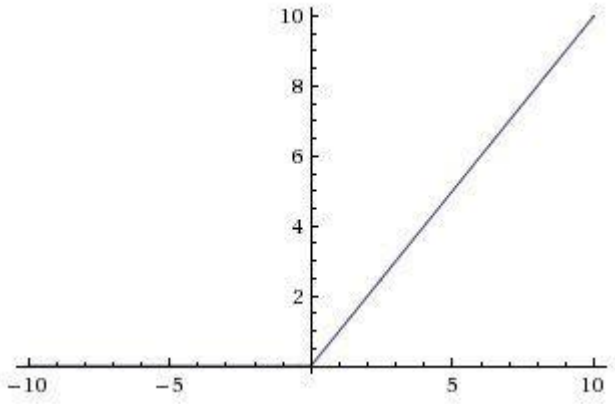


Activation function ReLU

- Computes f(x) = max(0,x)
- Does not saturate (in +region)

- Very computationally efficient

- Converges much faster than sigmoid/tanh in practice (e.g. 6x)



- **ReLU**

(Rectified Linear Unit)

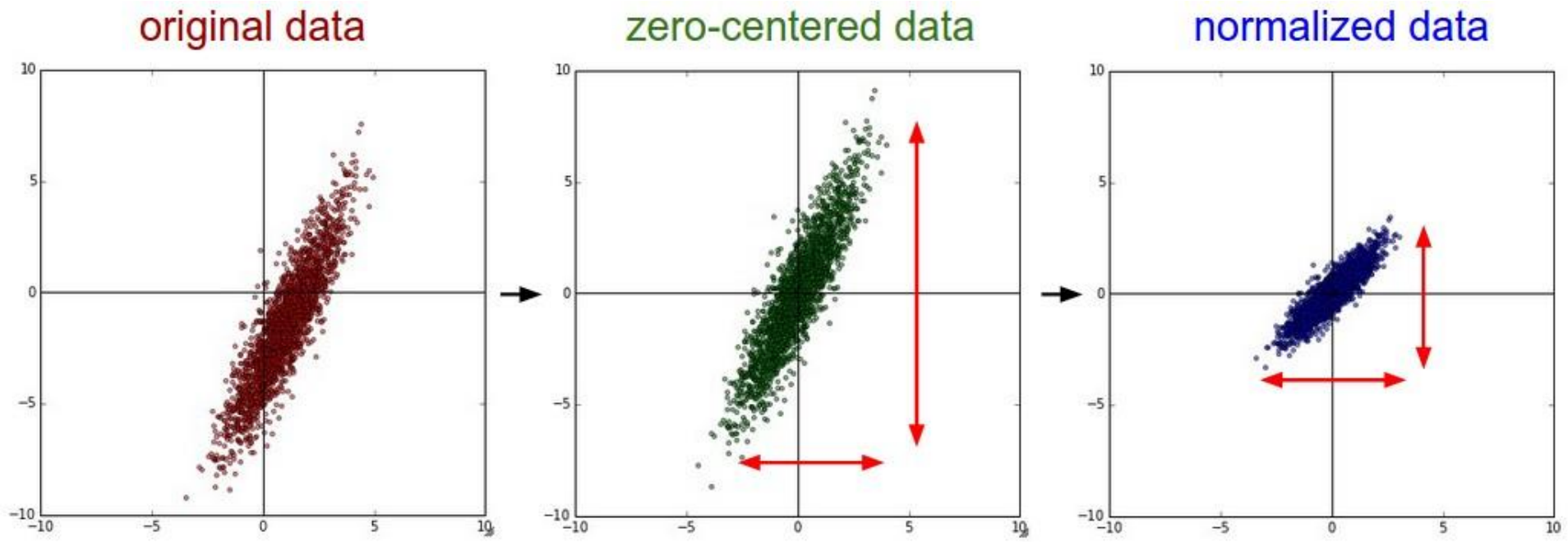
Mini-batch SGD

Loop:

1. Sample a batch of data
2. Forward prop it through the graph, get loss
3. Backprop to calculate the gradients
4. Update the parameters using the gradient

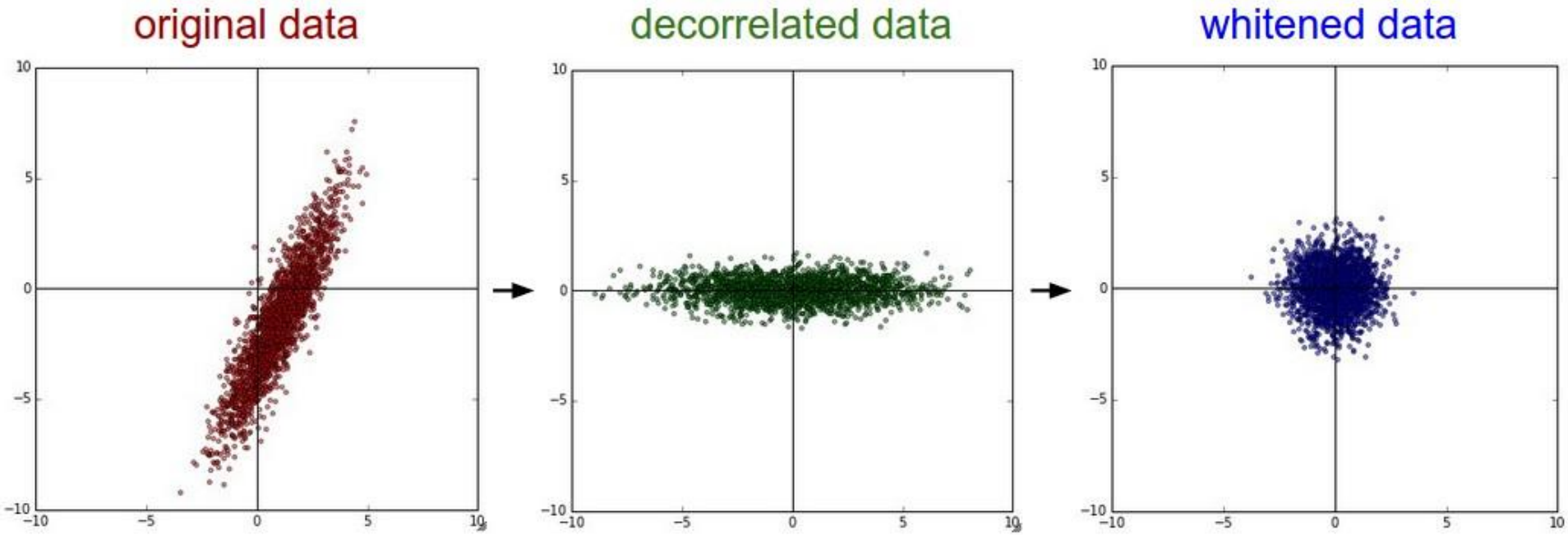
Data Pre-processing

- There are three common forms of data pre-processing a data matrix X, where we will assume that X is of size [N x D] (N is the number of data, D is their dimensionality).
- Mean subtraction: subtracting the mean across every individual feature in the data -- centring the cloud of data around the origin along every dimension.
- Normalization refers to normalizing the data dimensions so that they are of approximately the same scale.
  - divide each dimension by its standard deviation, once it has been zero-centred
  - normalizes each dimension so that the min and max along the dimension is -1 and 1 respectively


original data
zero-centered data
normalized data

PCA and Whitening

- PCA
- Whitening: The whitening operation takes the data in the eigenbasis and divides every dimension by the eigenvalue to normalize the scale.



Weight Initialization