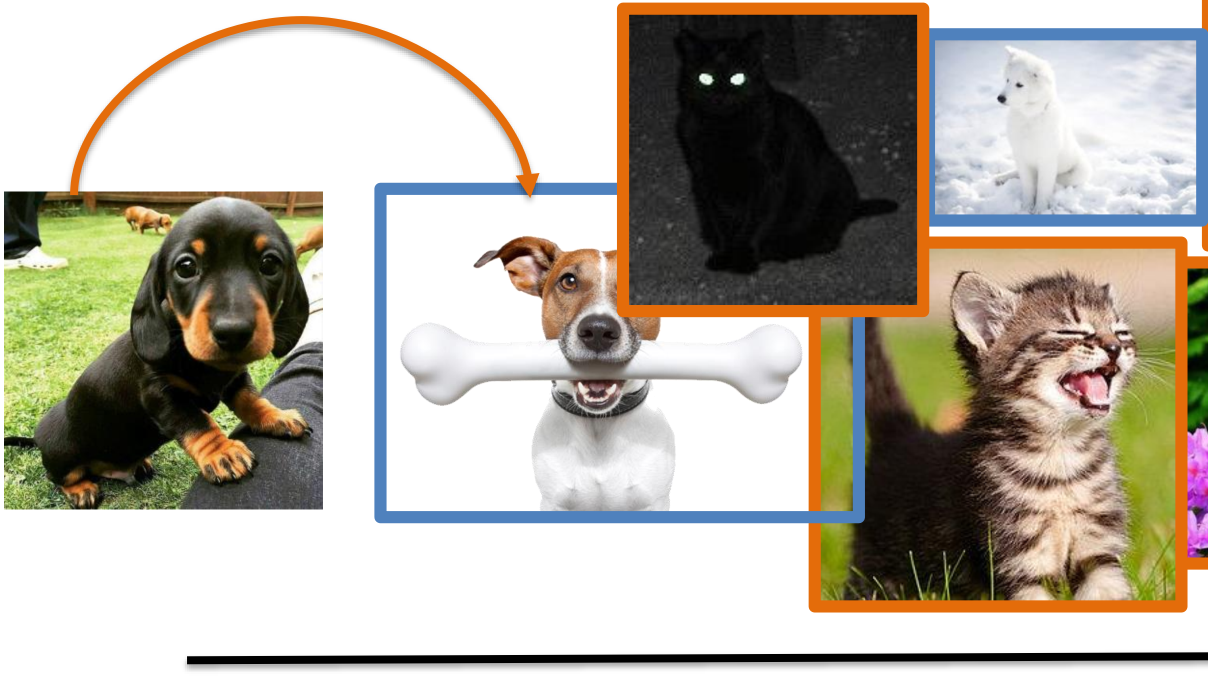
Lecture 2 Recap

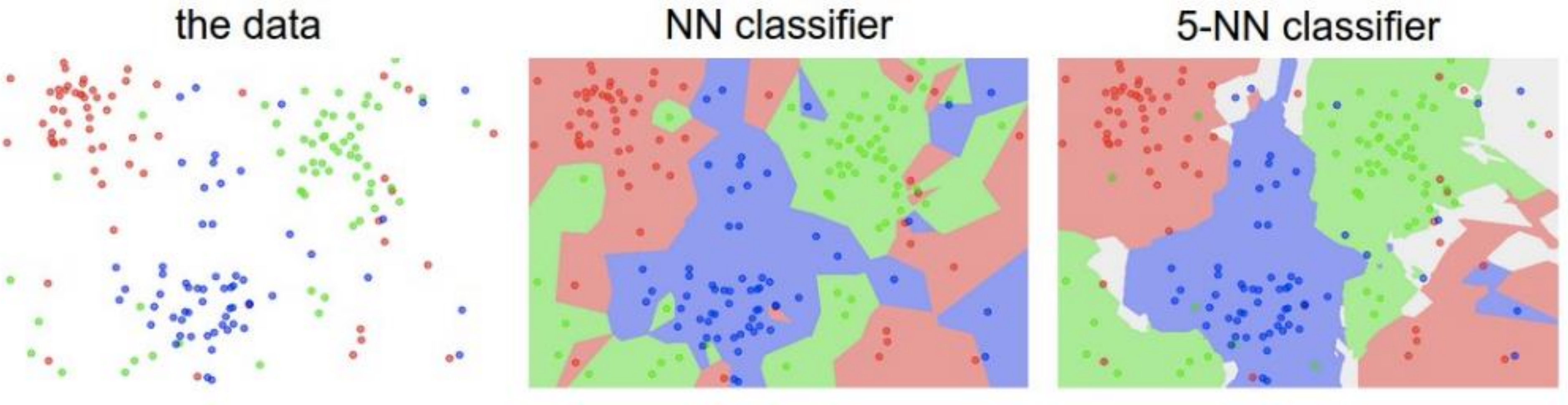
Nearest Neighbor

NN classifier = dog




distance

Nearest Neighbor



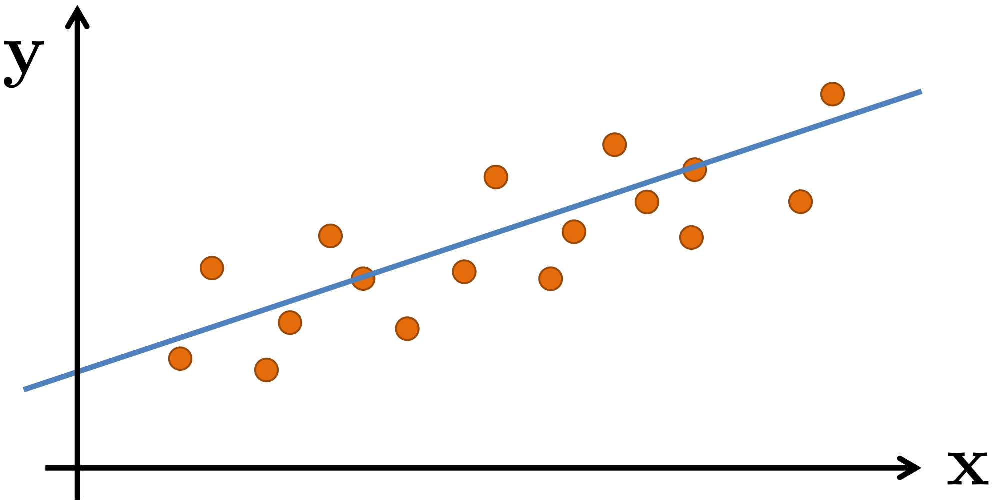
What is the performance on training data for NN classifier?

What classifier is more likely to perform best on test data?

Courtesy of Stanford course cs231n

Linear Regression

- Supervised learning
- Find a linear model that explains a target $\mathbf{y}$ given the inputs $\mathbf{X}$

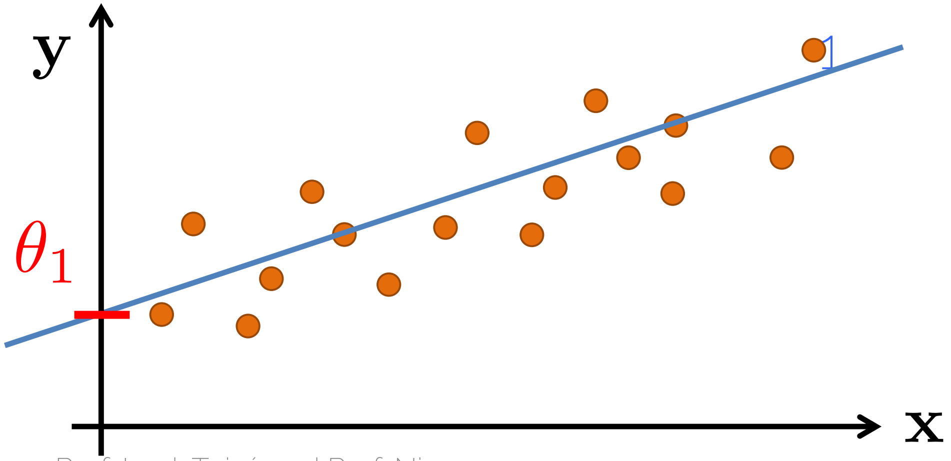


Linear Regression

- A linear model is expressed in the form

$$\hat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j = x_{i1}\boxed{\theta_1} + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$

bias



Logistic Regression

- Loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

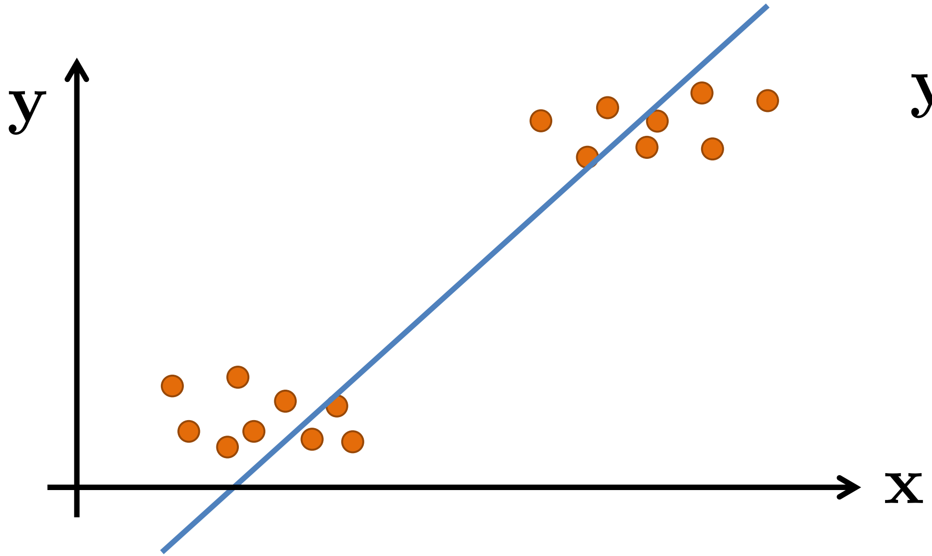
- Cost function

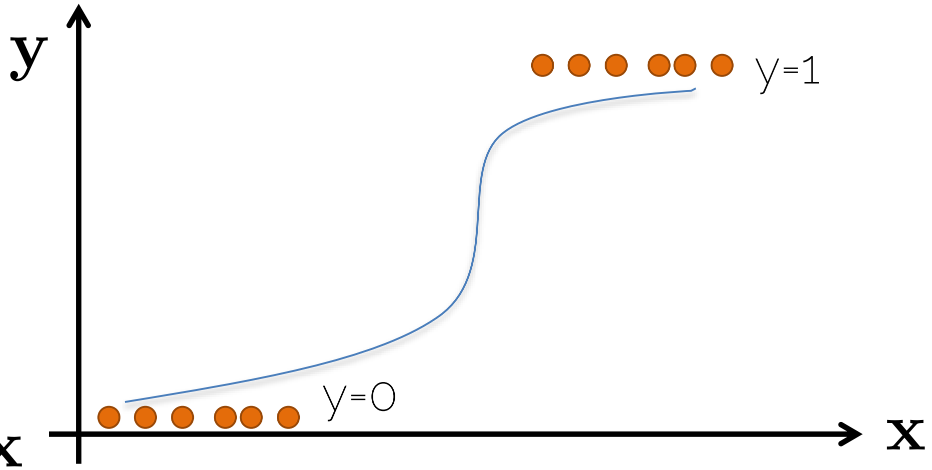
$$C(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Minimization

$$\hat{y}_i = \sigma(\mathbf{x}_i \boldsymbol{\theta})$$

Linear vs Logistic Regression





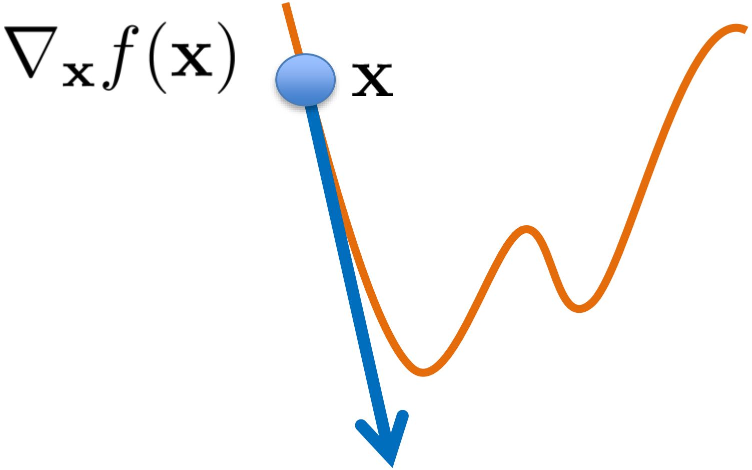
Prediction can exceed
training sample range
-> in case of classification [0;1] that's a real issue

Prediction is guaranteed
to be within 0 and 1

(Vanilla) Gradient Descent

$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

Learning rate



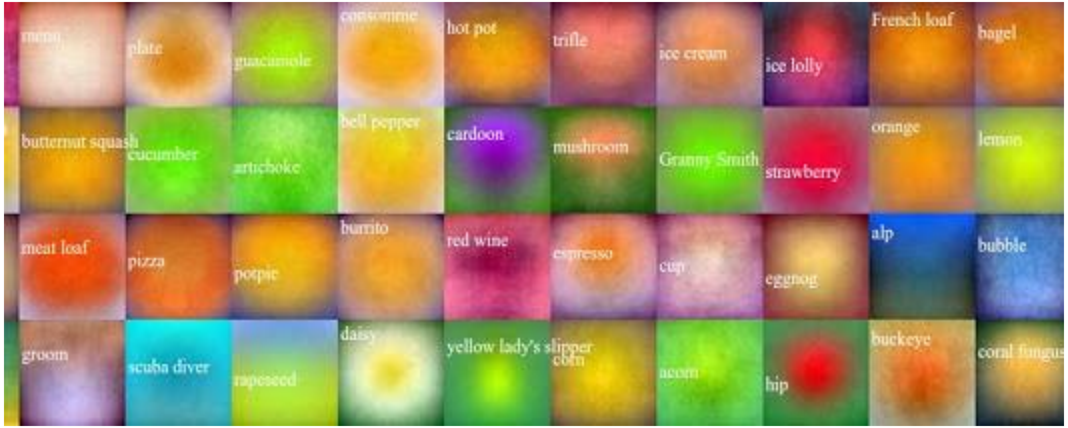
$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$

Introduction to Neural Networks

Neural Network

- Linear score function $f = Wx$

On CIFAR-10



Credit: Li/Karpathy/Johnson

On ImageNet

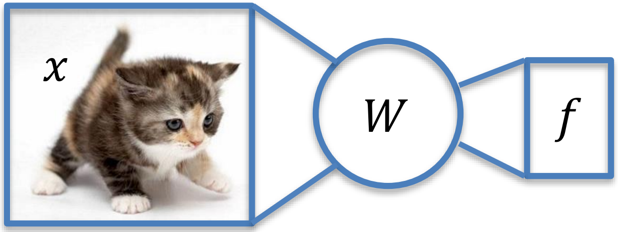
Neural Network

- Linear score function $f = Wx$

- Neural network is a nesting of 'functions'
  - 2-layers: $f = W_2 \max(0, W_1 x)$
  - 3-layers: $f = W_3 \max(0, W_2 \max(0, W_1 x))$
  - 4-layers: $f = W_4 \tanh\left(\mathrm{W}_3, \max(0, W_2 \max(0, W_1 x))\right)$
  - 5-layers: $f = W_5 \sigma(W_4 \tanh(\mathrm{W}_3, \max(0, W_2 \max(0, W_1 x))))$
  - … up to hundreds of layers

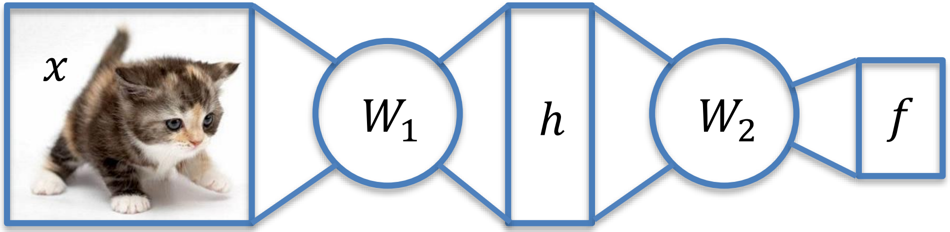
Neural Network

1-layer network: $f = Wx$



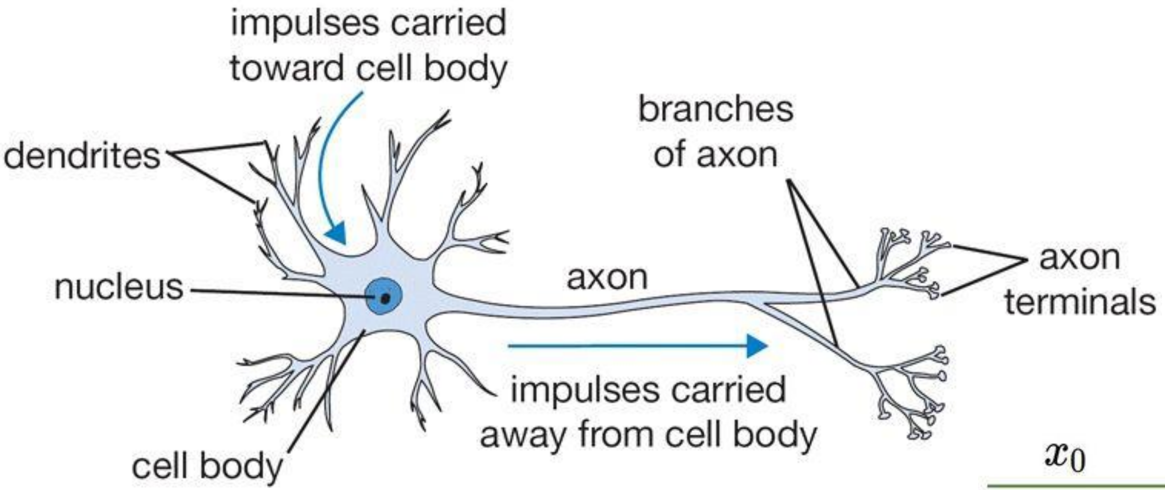
$128 \times 128 = 16384$ $\quad$ 10

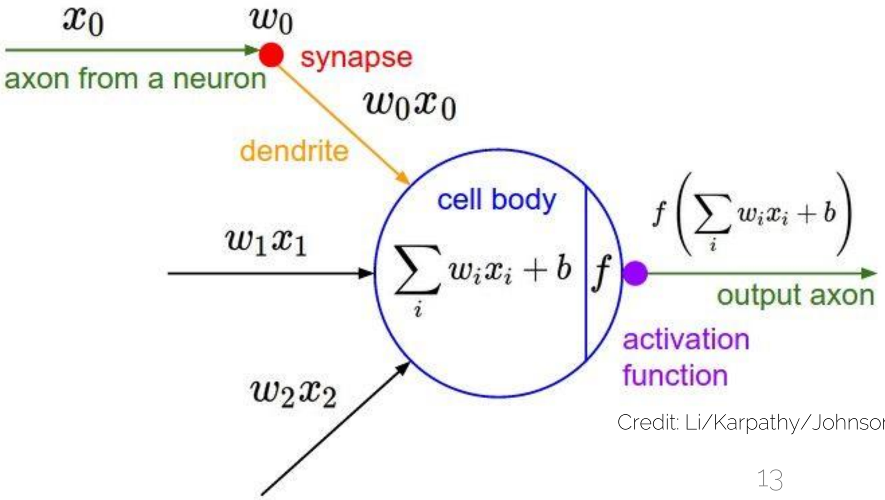
2-layer network: $f = W_2 \max(0, W_1 x)$



$128 \times 128 = 16384$ $\quad$ 1000 $\quad$ 10

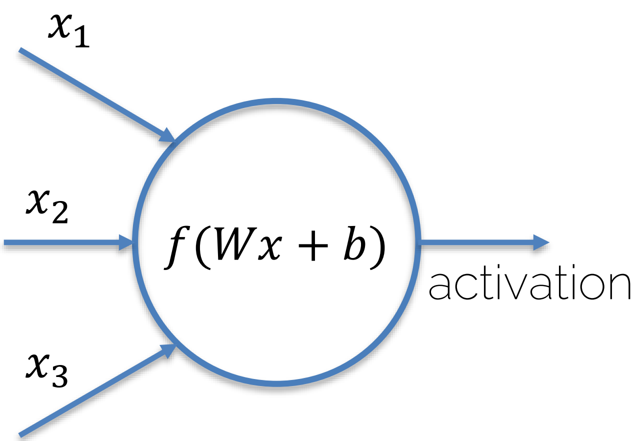
Neurons





Credit: Li/Karpathy/Johnson

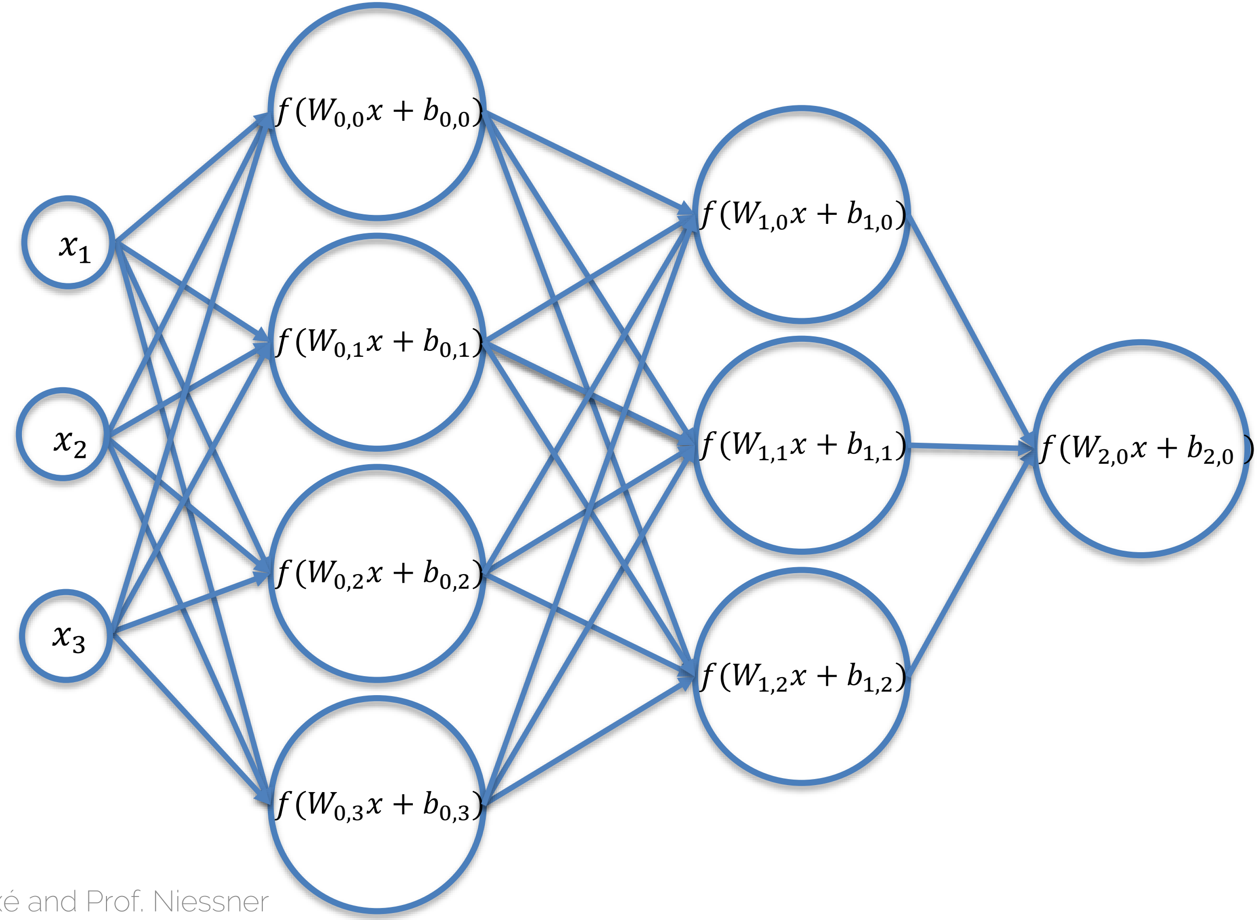
Neurons



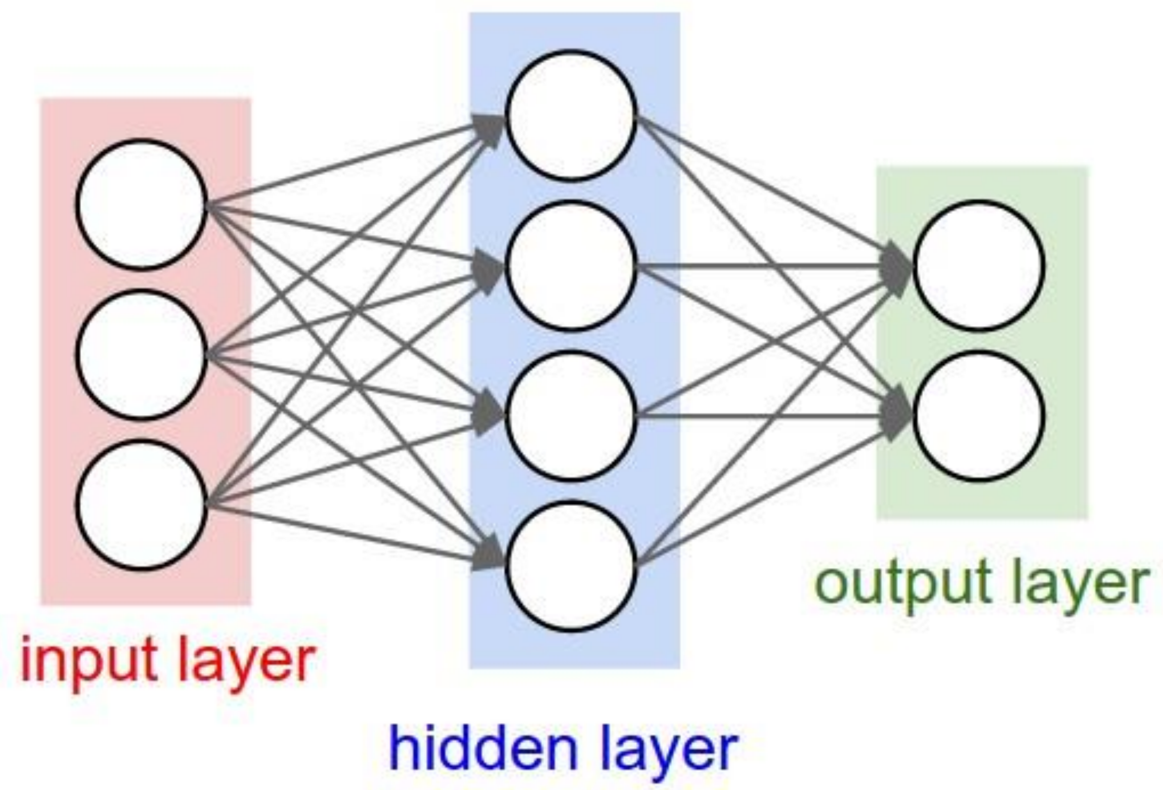
Linear function: $Wx + b$
Non-linearity (activation: $f(x)$

Every neuron computes: $f(Wx + b)$

Net of Neurons

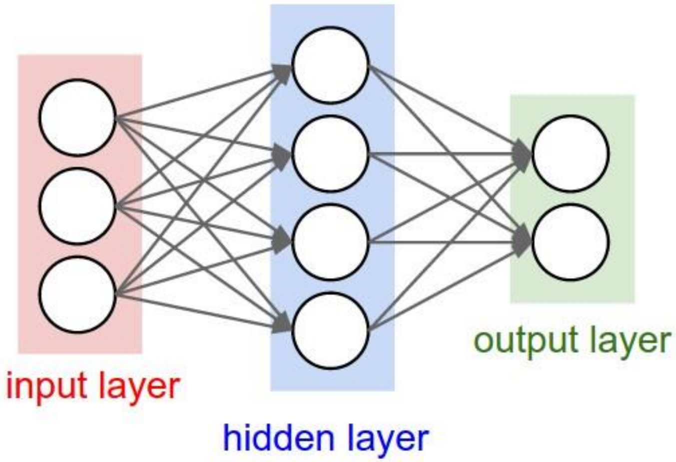


Neural Network

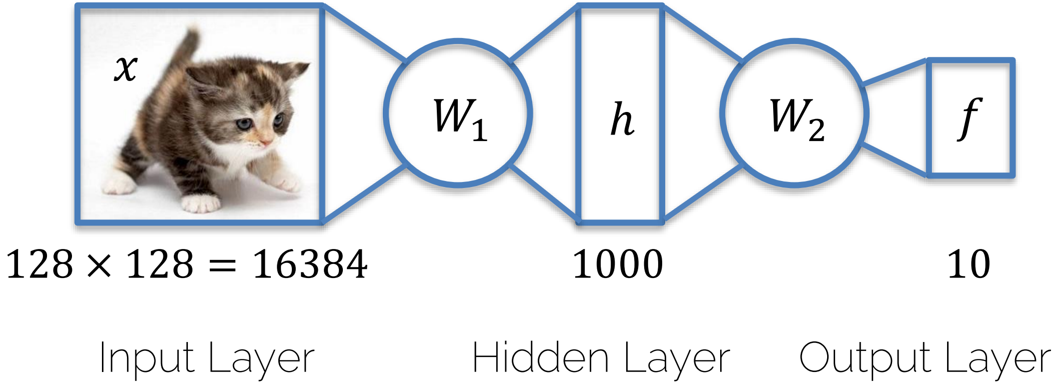


Credit: Li/Karpathy/Johnson

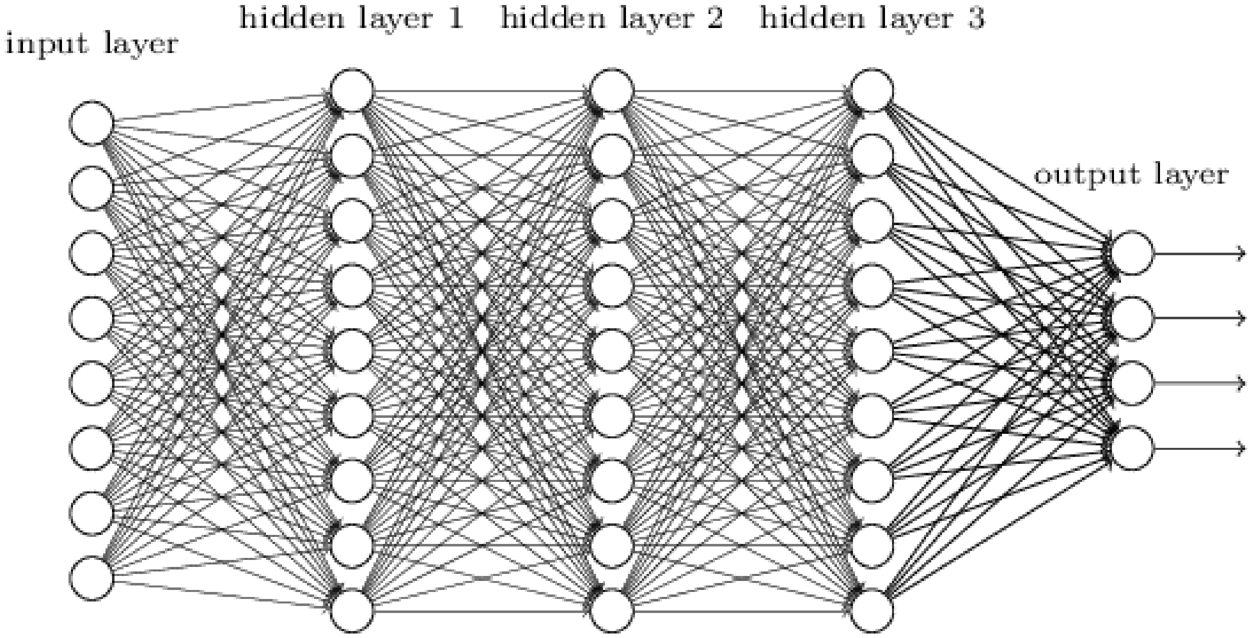
Neural Network



2-layer network: $f = W_2 \max(0, W_1 x)$



Neural Network



Neural Network

$$f = W_3 \cdot (W_2 \cdot (W_1 \cdot x))$$

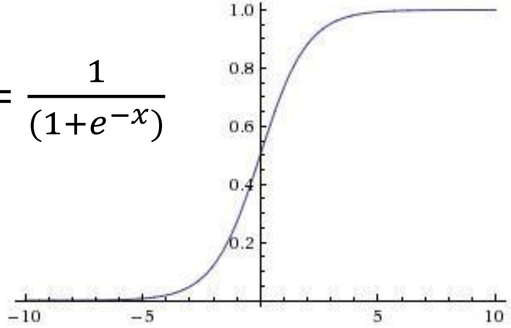
Why activation functions?

Why not just concatenate?

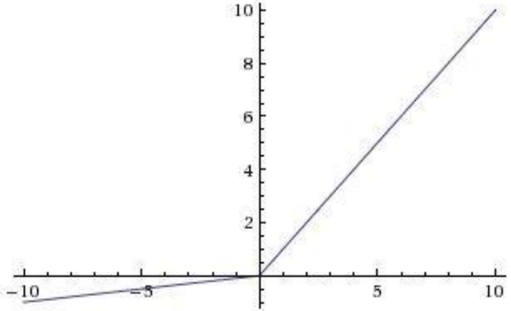
Would be much cheaper to compute….

Activation Functions

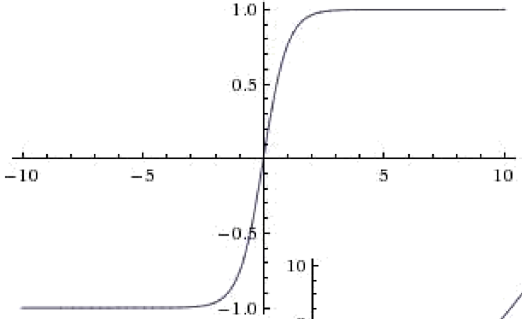
Sigmoid: $\boldsymbol{\sigma(x) = \frac{1}{(1+e^{-x})}}$



Leaky ReLU: $\mathbf{max}(0.1x, x)$



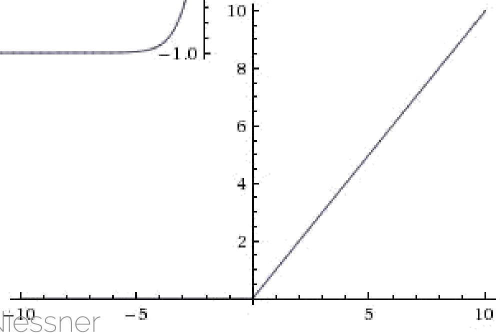
tanh: $\mathbf{tanh}(x)$



Parametric ReLU: $\mathbf{max}(\alpha x, x)$

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

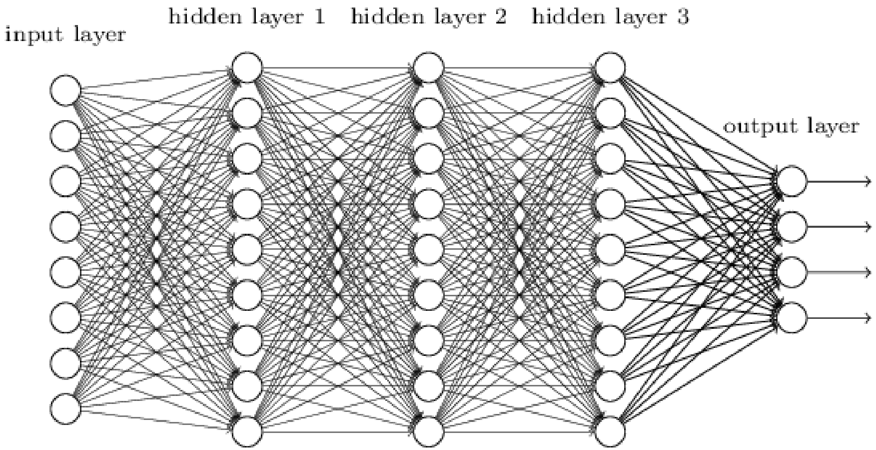
ReLU: $\mathbf{max}(0, x)$



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

Neural Network

Why organize a neural network into layers?



Neural Network

- Summary
  - Given a dataset with ground truth training pairs $[x_i; y_i]$,
  - Find optimal weights $\boldsymbol{W}$ using stochastic gradient descent, such that the loss function is minimized
  - Compute gradients with backpropagation (use batch-mode; more later)
  - Iterate many times over training set (SGD; more later)

Artificial Neural Network vs Brain

Artificial neural networks: inspired but not even close to the brain!
It's much more complex than simple linearity + activations
Great for the media and news articles ☺

Artificial Neural Network vs Brain

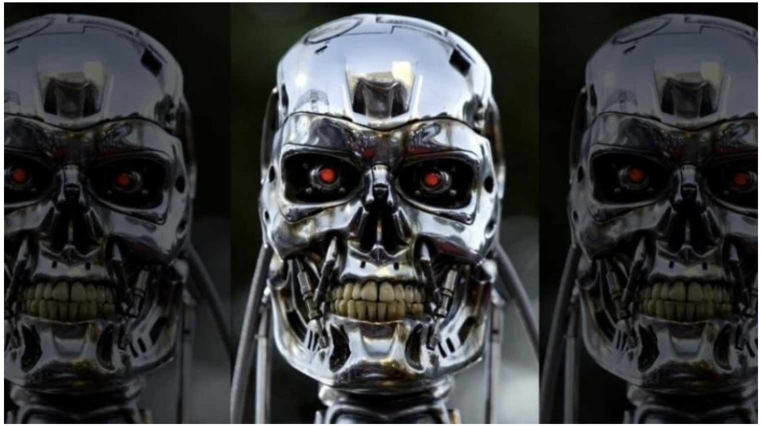
FOX NEWS .com

U.S. World Opinion Politics Entertainment Business Lifestyle TV Radio More Login Wa

Hot Topics 'Unprecedented' NoKo threat | Another ESPN controversy | Serial killer scare

GOOGLE · 3 days ago

Google's artificial intelligence computer 'no longer constrained by limits of human knowledge'



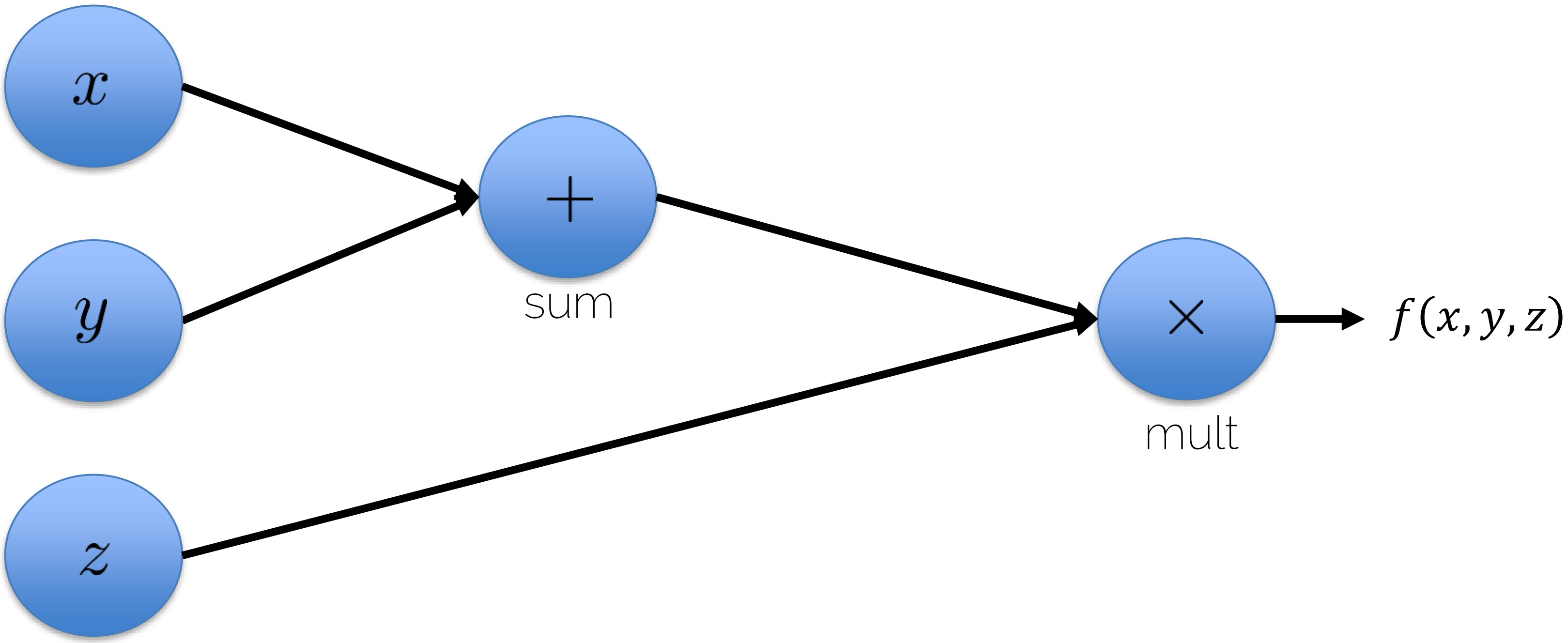
Computational Graphs

Computational Graphs

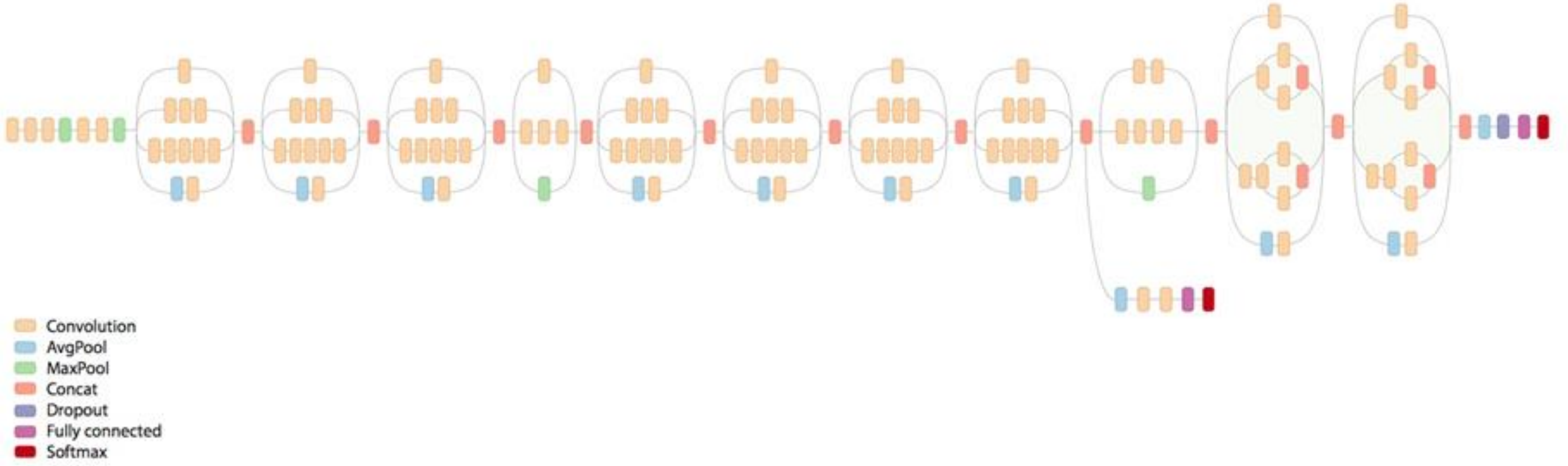
- Neural network is a computational graph
  - It has compute nodes
  - It has edges that connect nodes
  - It is directional
  - It is organized in 'layers'

Computational Graphs

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



Computational Graphs

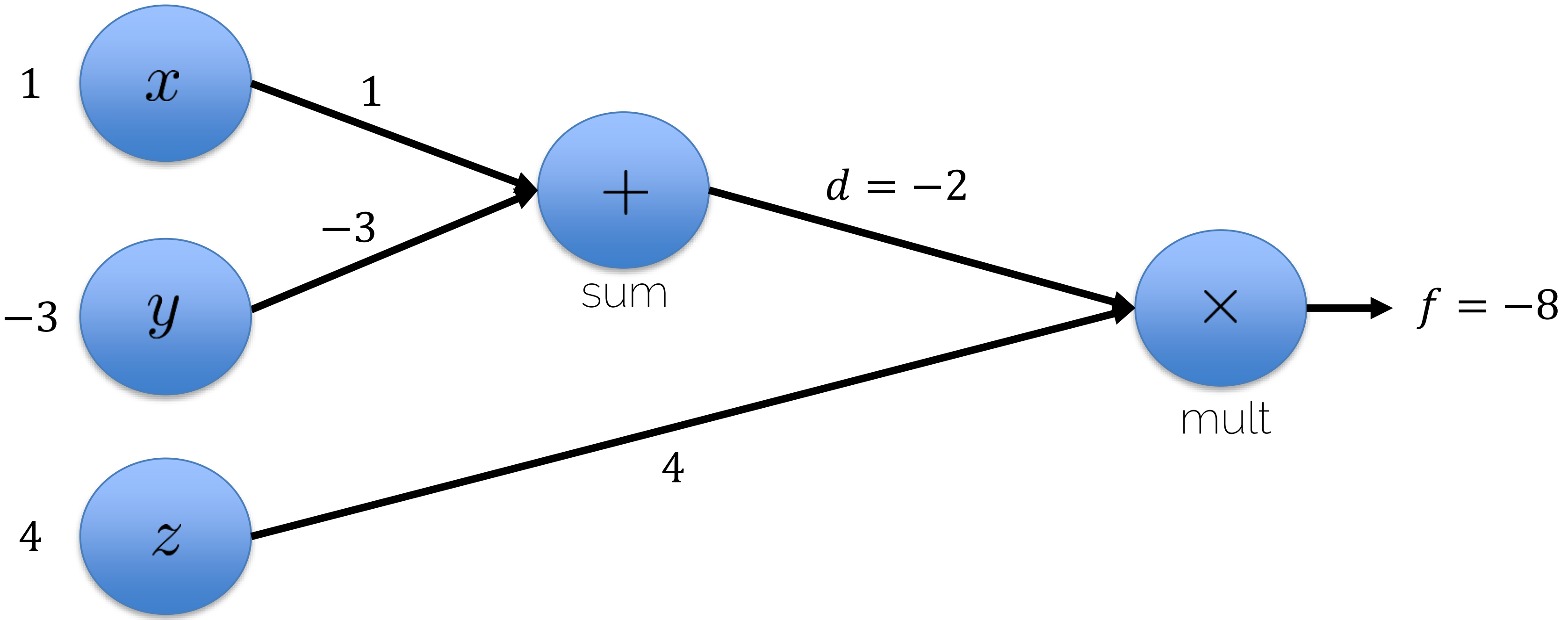


Another view of GoogLeNet's architecture.

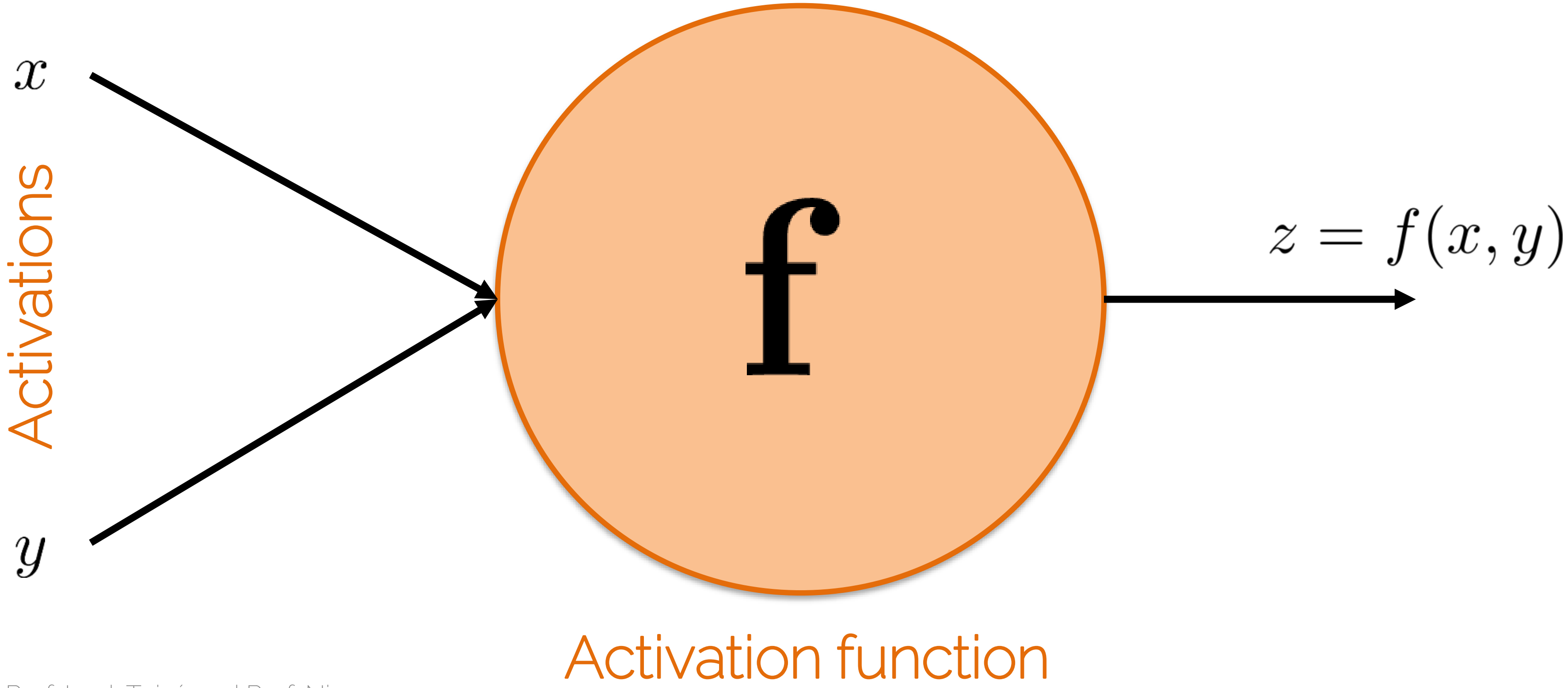
Evaluation: Forward Pass

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

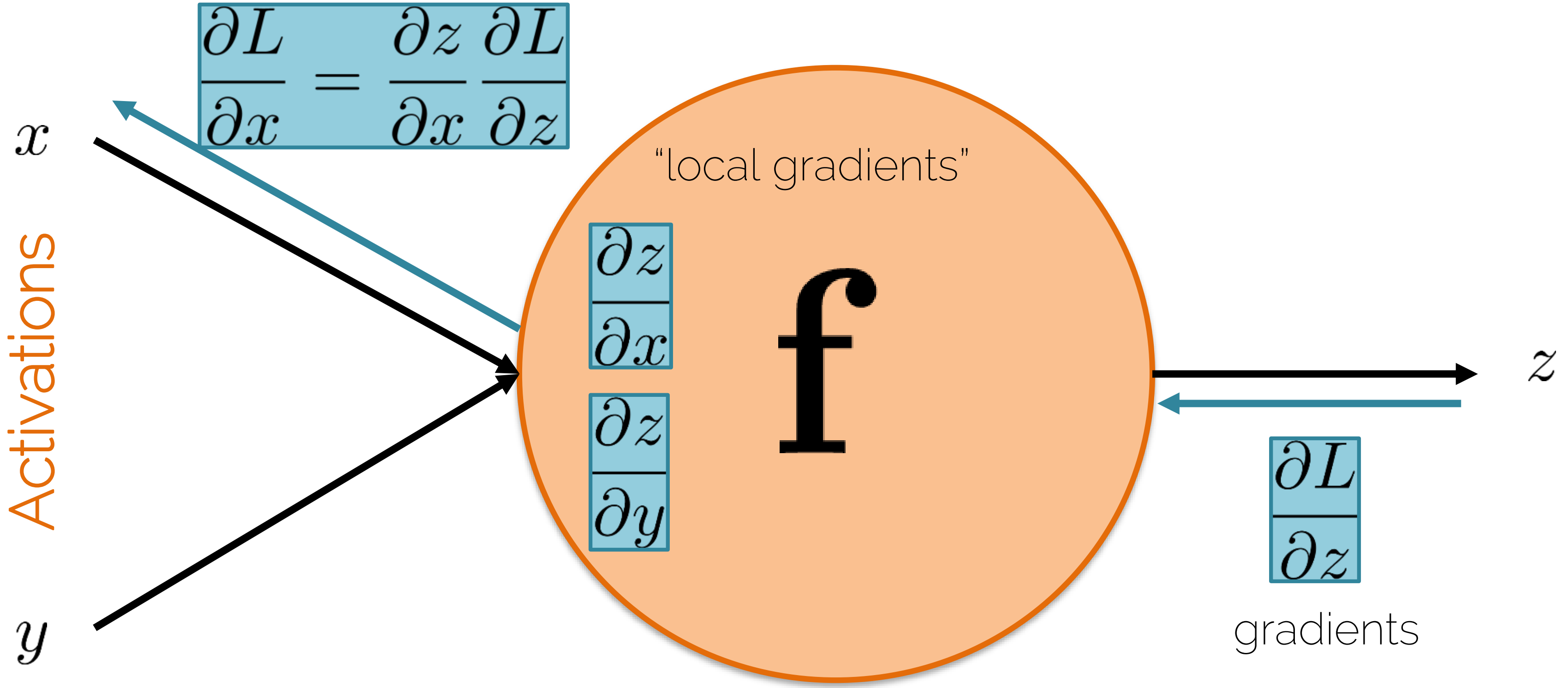
Initialization $x = 1, y = -3, z = 4$



The Flow of Gradients



The Flow of Gradients



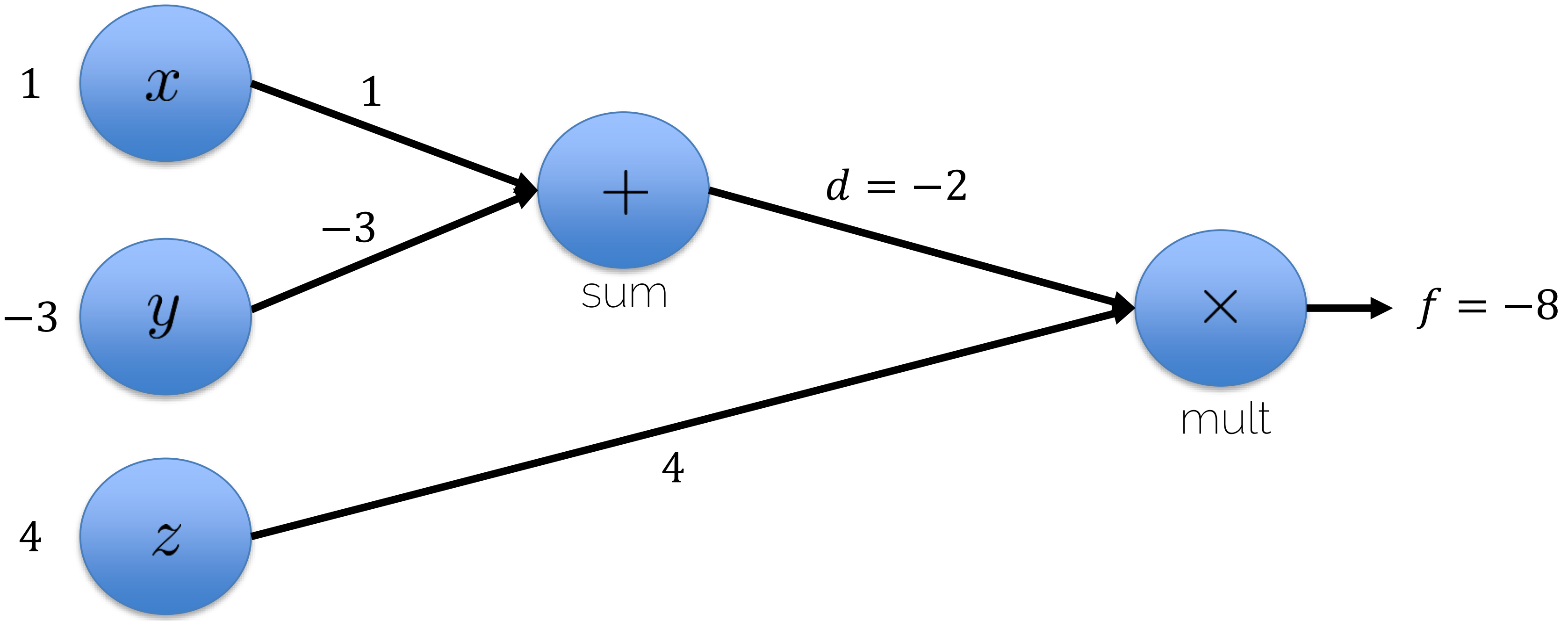
Activation function

Backpropagation

Backprop: Forward Pass

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

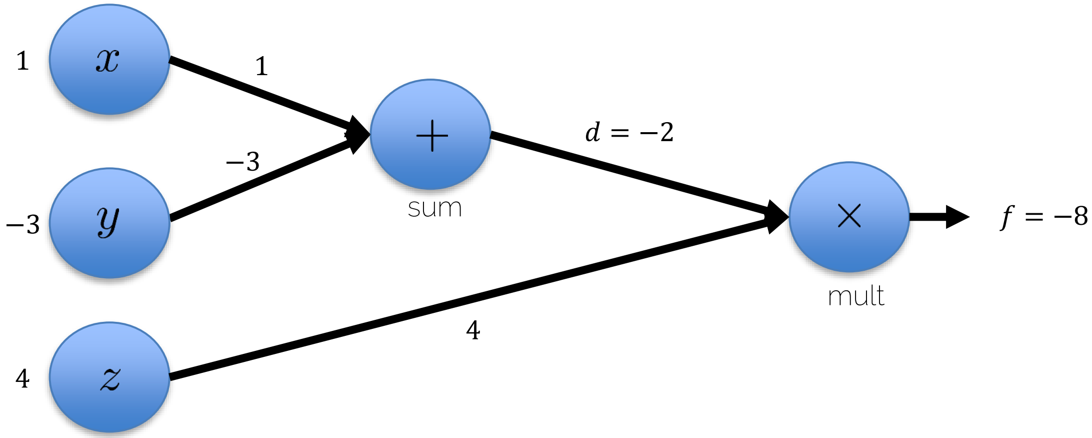
Initialization $x = 1, y = -3, z = 4$



Backprop: Backward Pass

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

with $x = 1, y = -3, z = 4$



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

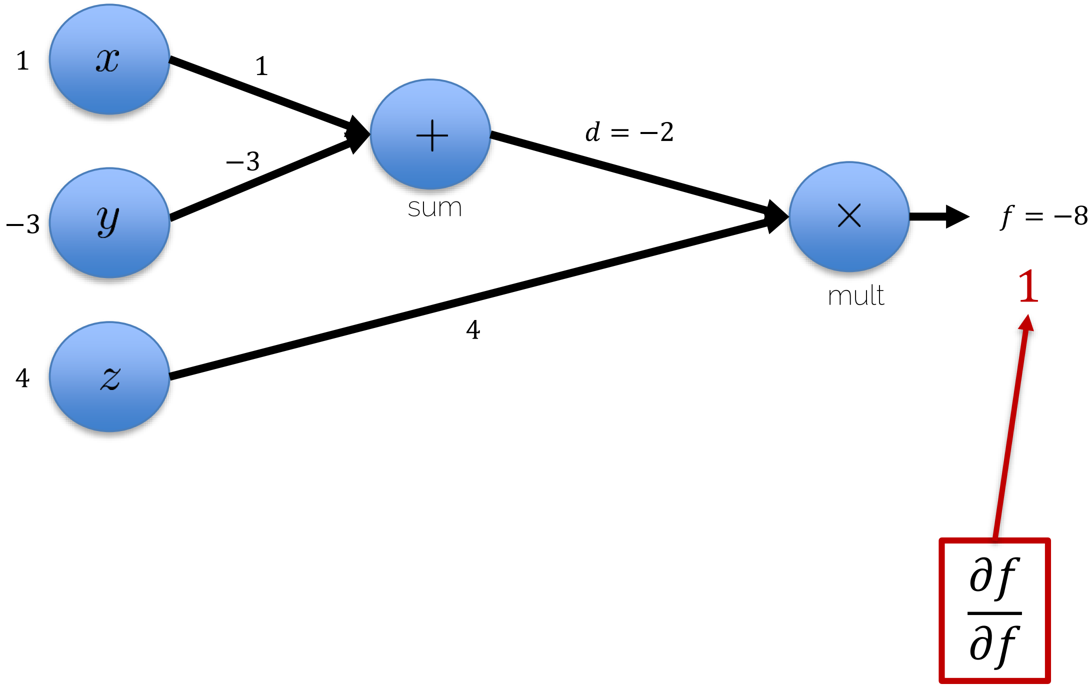
$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Backprop: Backward Pass

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

with $x = 1, y = -3, z = 4$



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

$f = d \cdot z$ $\qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Backprop: Backward Pass

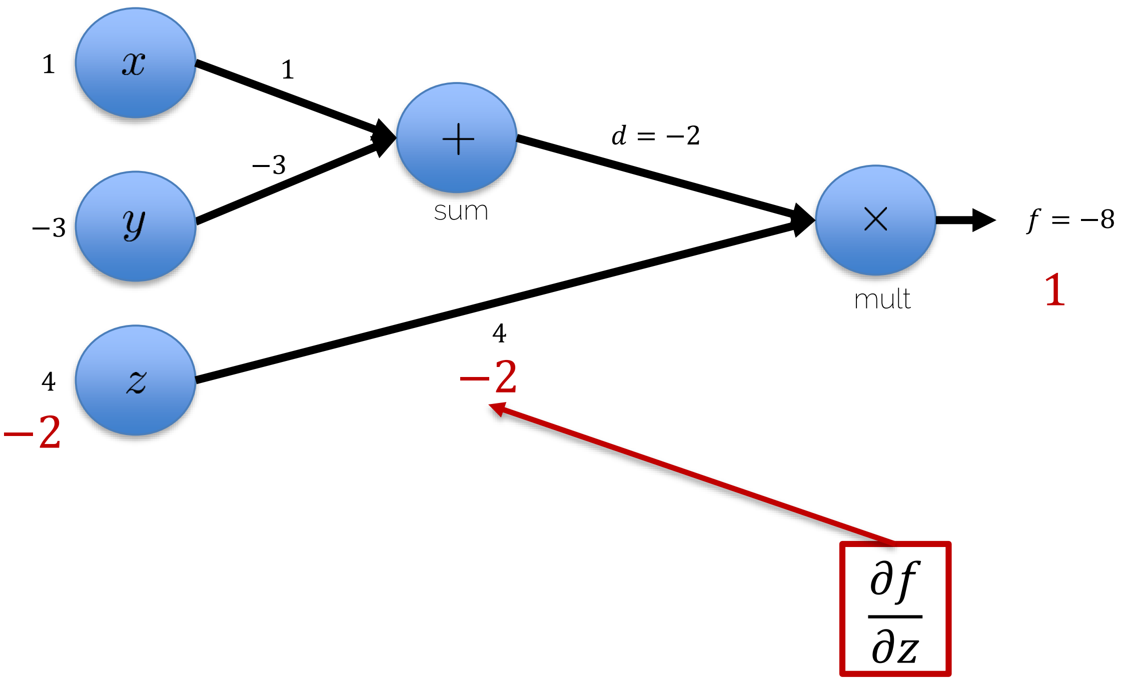
$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$

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

$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \boxed{\frac{\partial f}{\partial z} = d}$

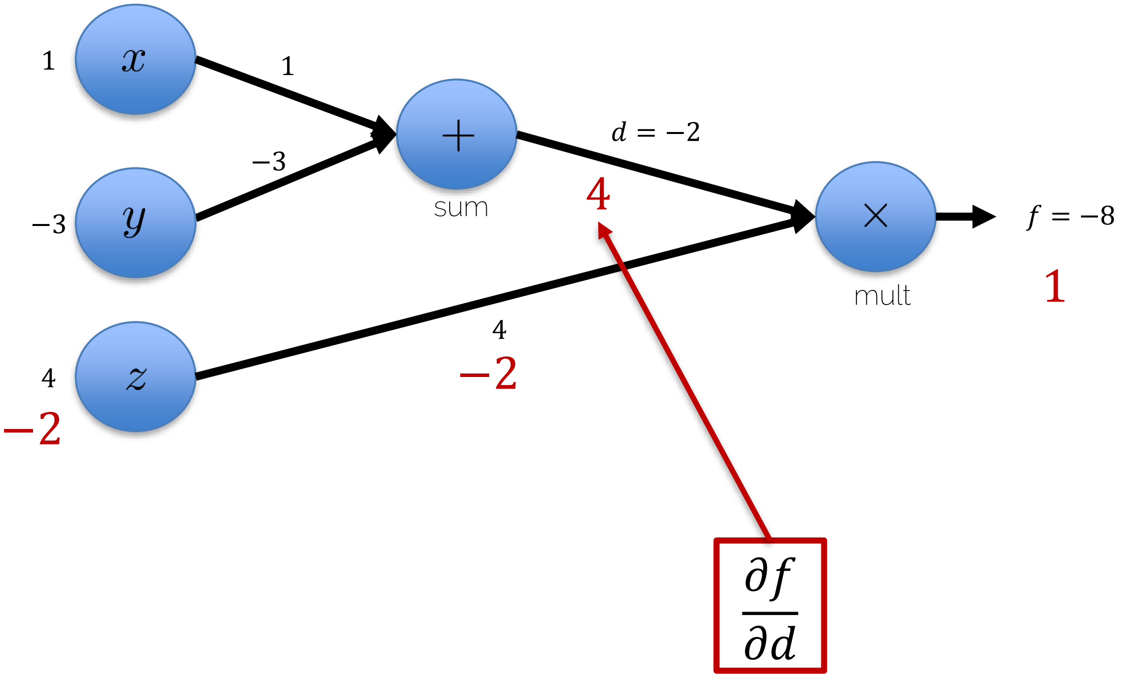
What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?



Backprop: Backward Pass

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

with $x = 1, y = -3, z = 4$



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

$f = d \cdot z \qquad \boxed{\frac{\partial f}{\partial d} = z}, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Backprop: Backward Pass

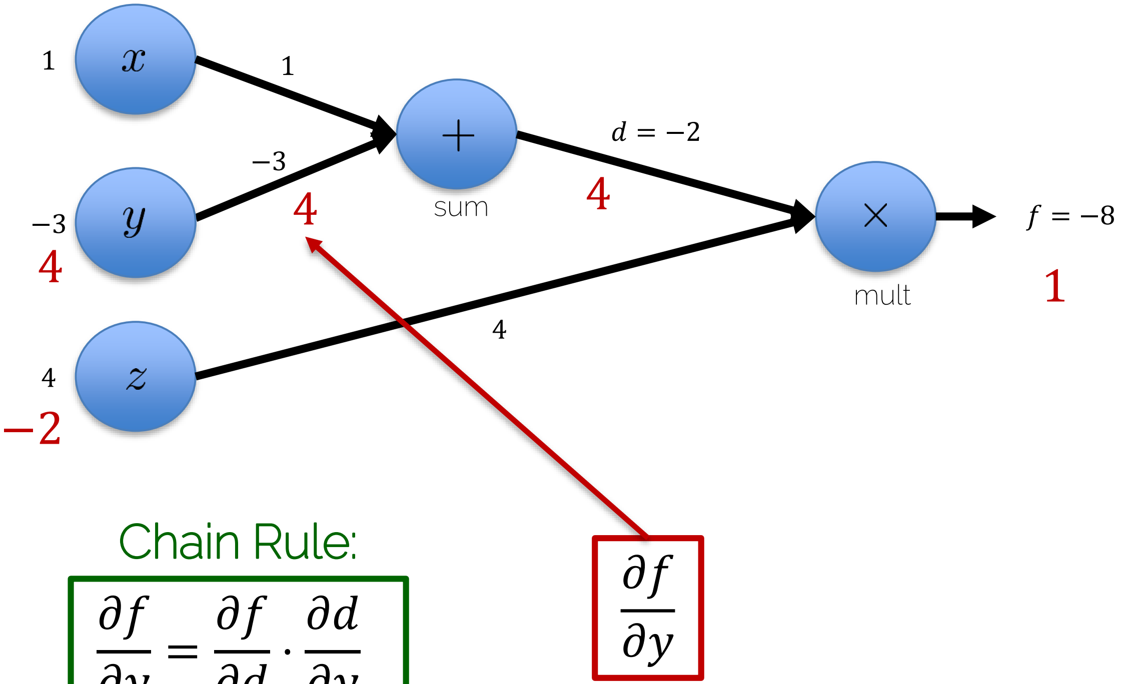
$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$

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

$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?



Chain Rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial y}}$$

$$\rightarrow \frac{\partial f}{\partial y} = 4 \cdot 1 = 4$$

Backprop: Backward Pass

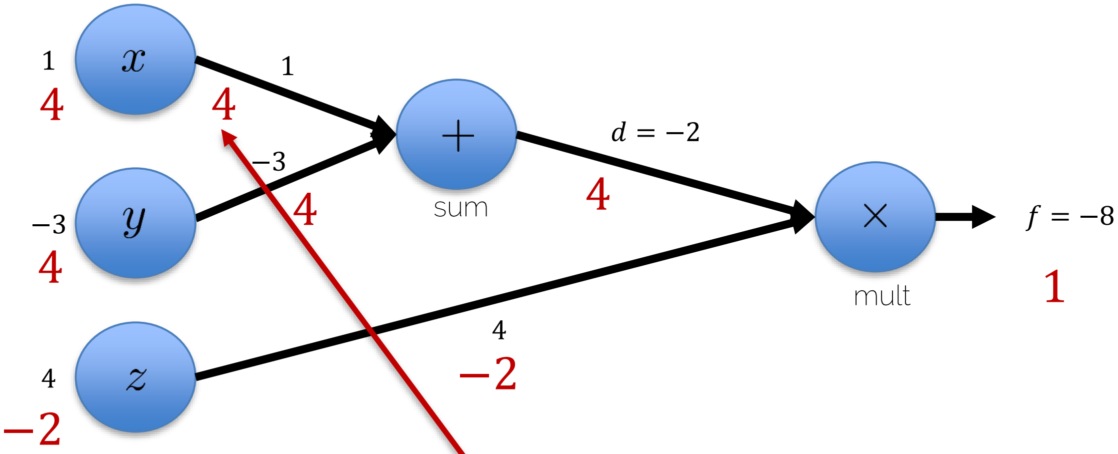
$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$

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

$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?



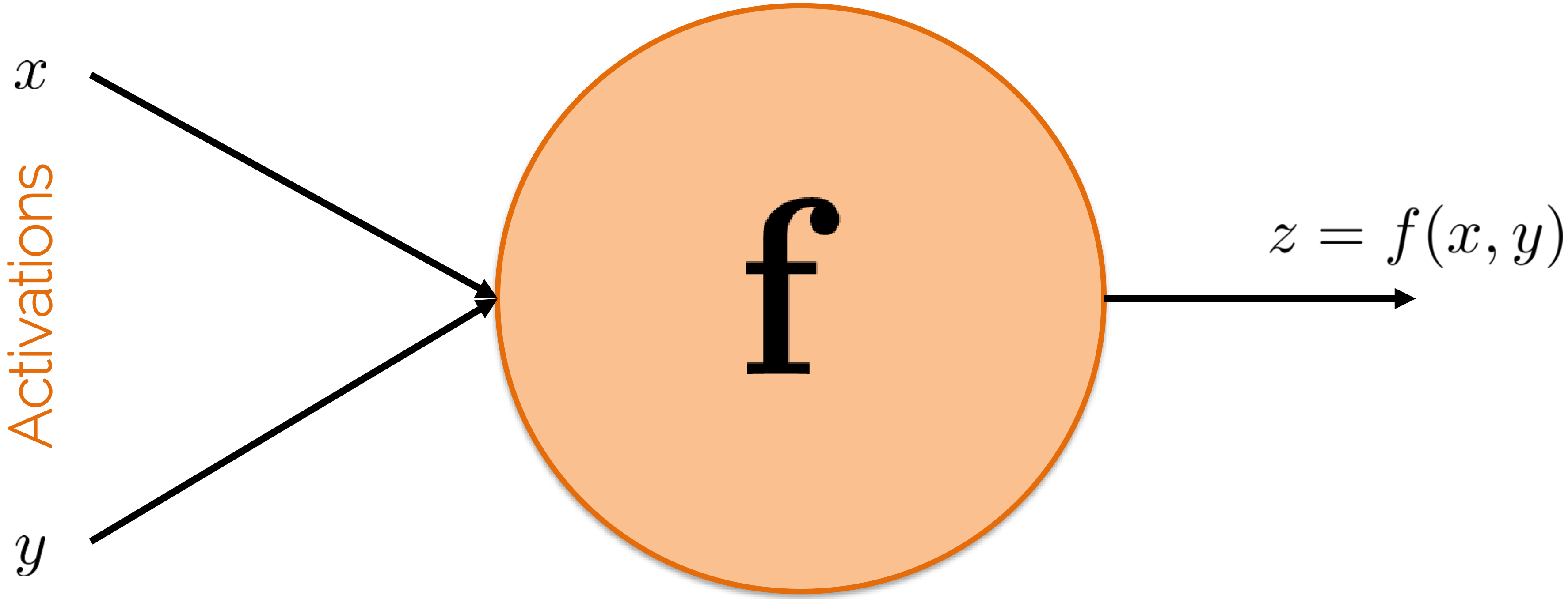
Chain Rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial x}}$$

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

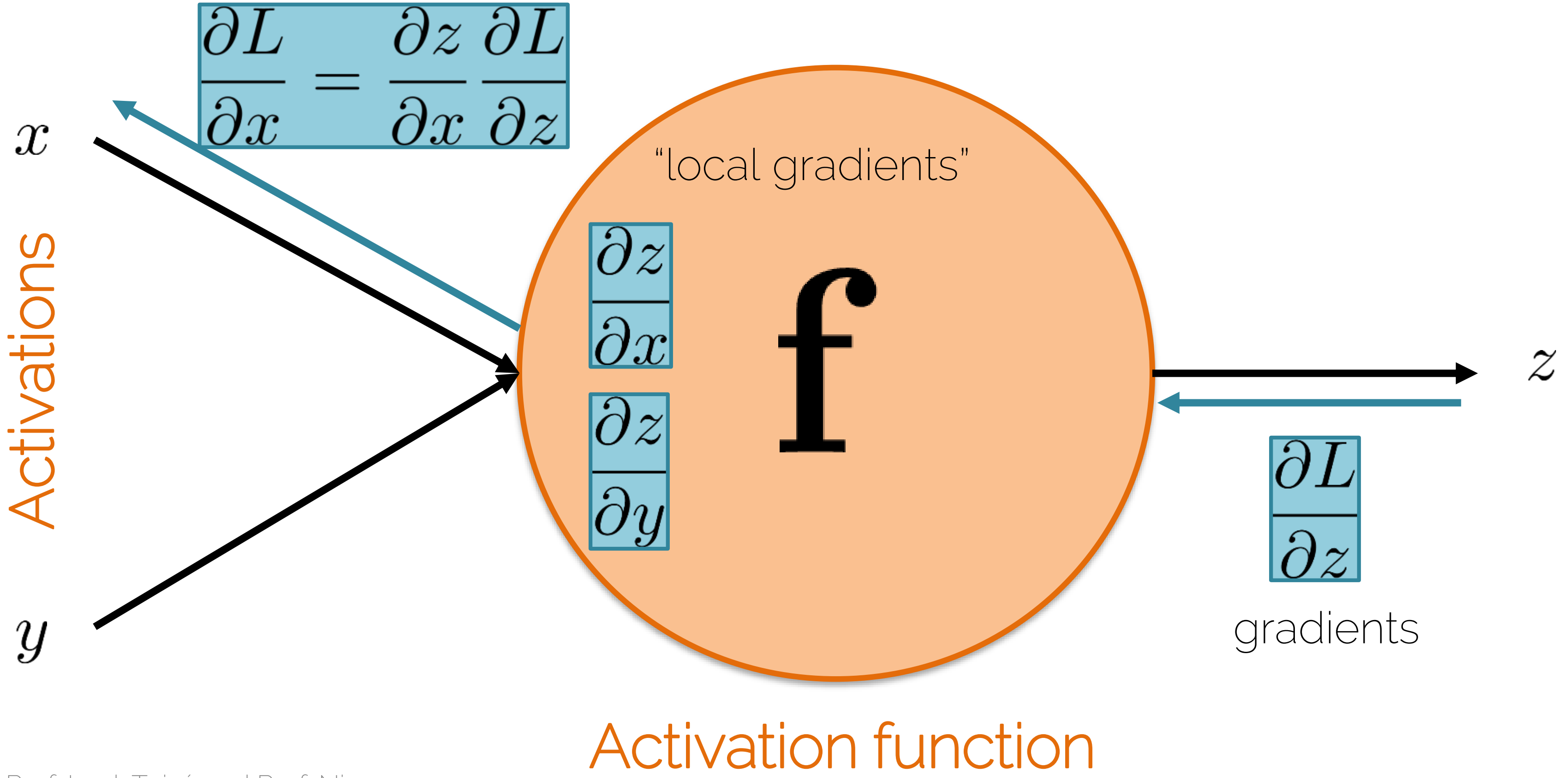
$$\rightarrow \frac{\partial f}{\partial x} = 4 \cdot 1 = 4$$

The Flow of Gradients



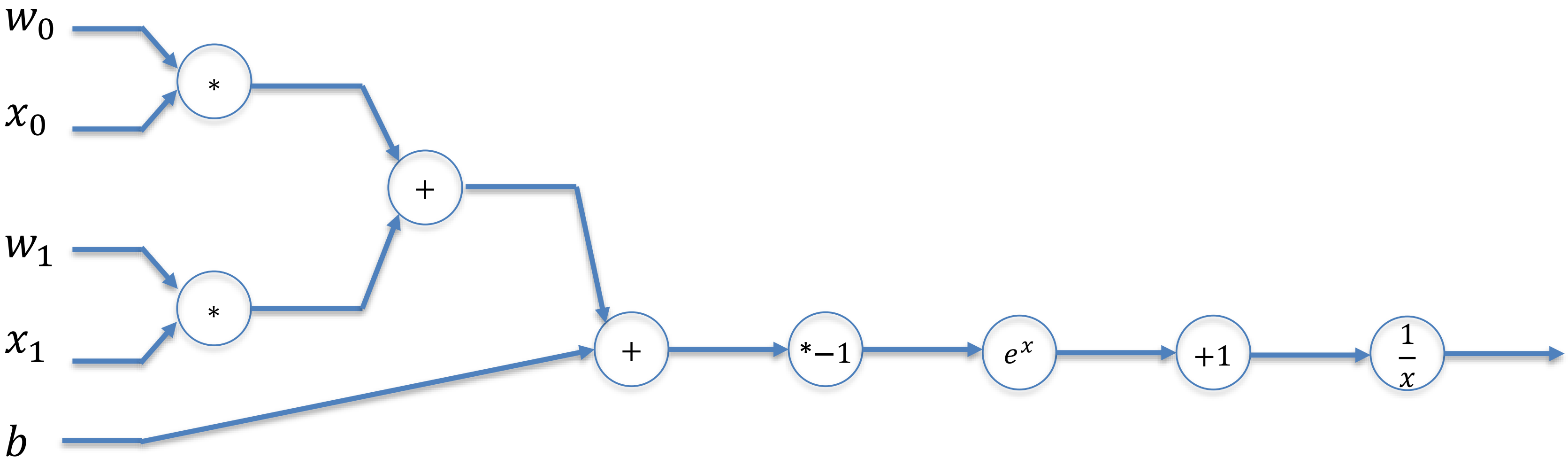
Activation function

The Flow of Gradients



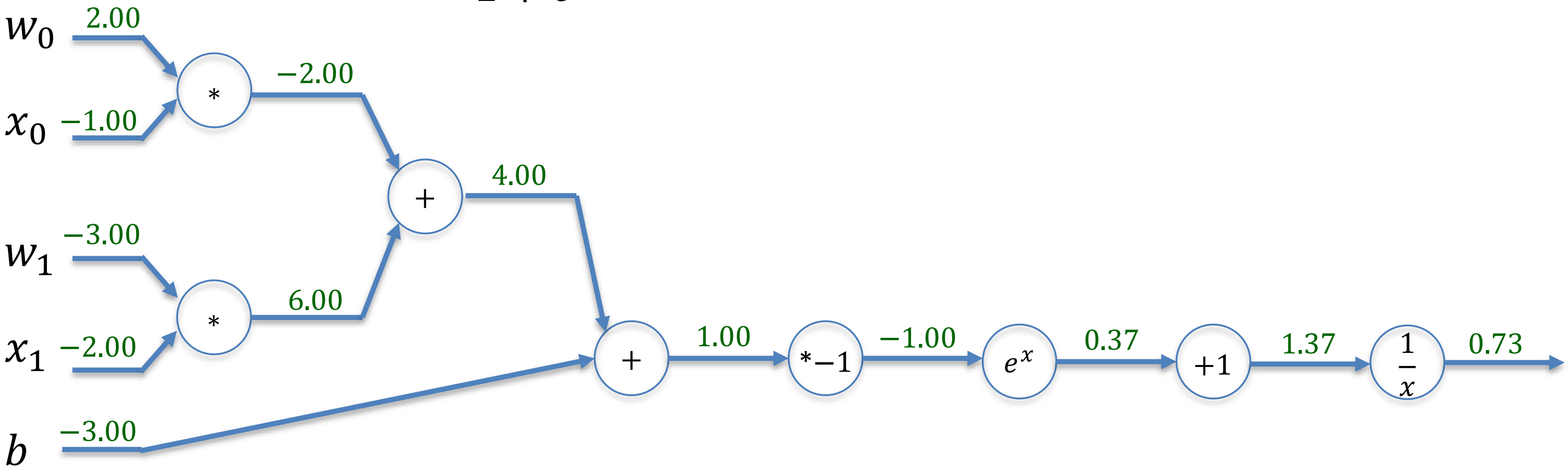
Backprop

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



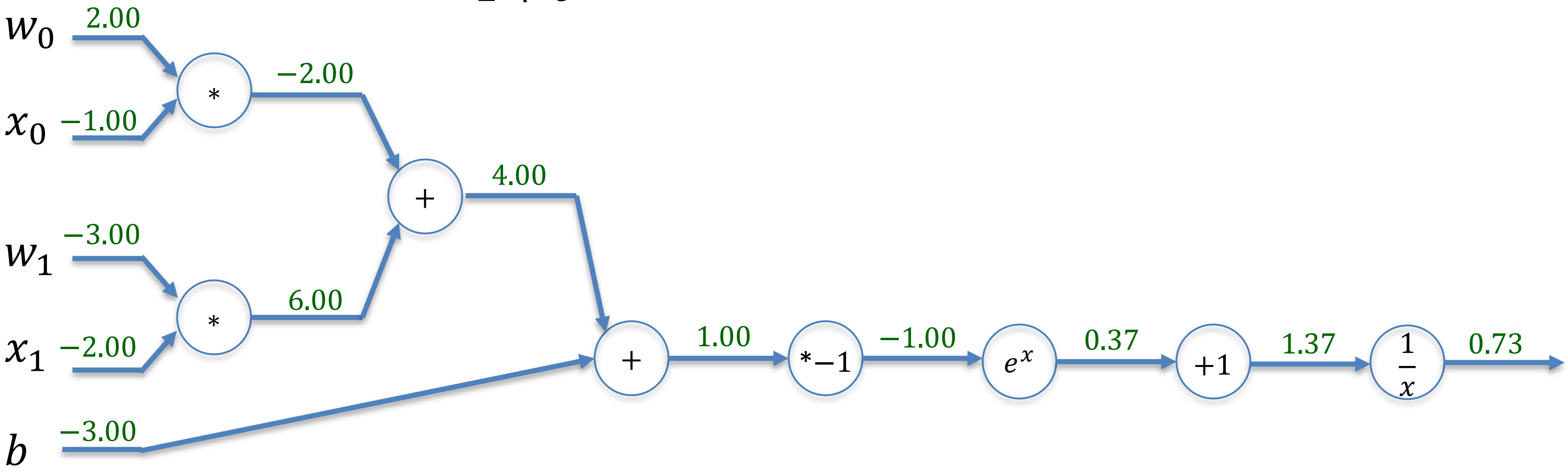
Backprop

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



Backprop

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



$$f(x) = e^x \quad \text{->} \quad \frac{\partial f}{\partial x} = e^x$$

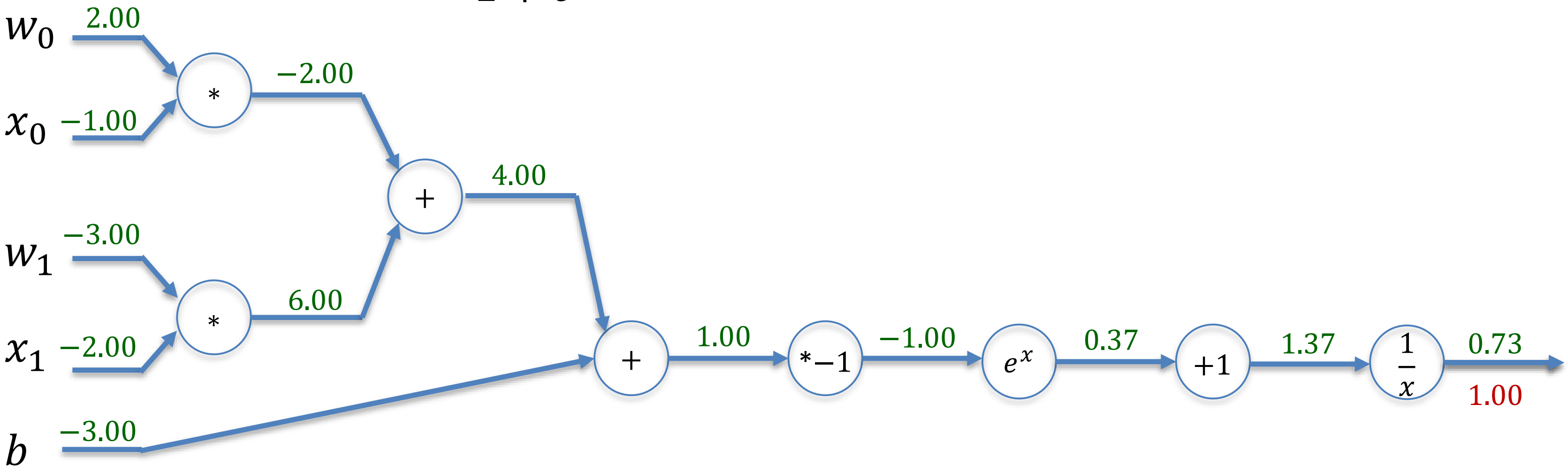
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

$$f(x) = \frac{1}{x} \quad \text{->} \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f_c(x) = c + x \quad \text{->} \quad \frac{\partial f_c}{\partial x} = 1$$

Backprop

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



$$f(x) = e^x \quad -> \quad \frac{\partial f}{\partial x} = e^x$$

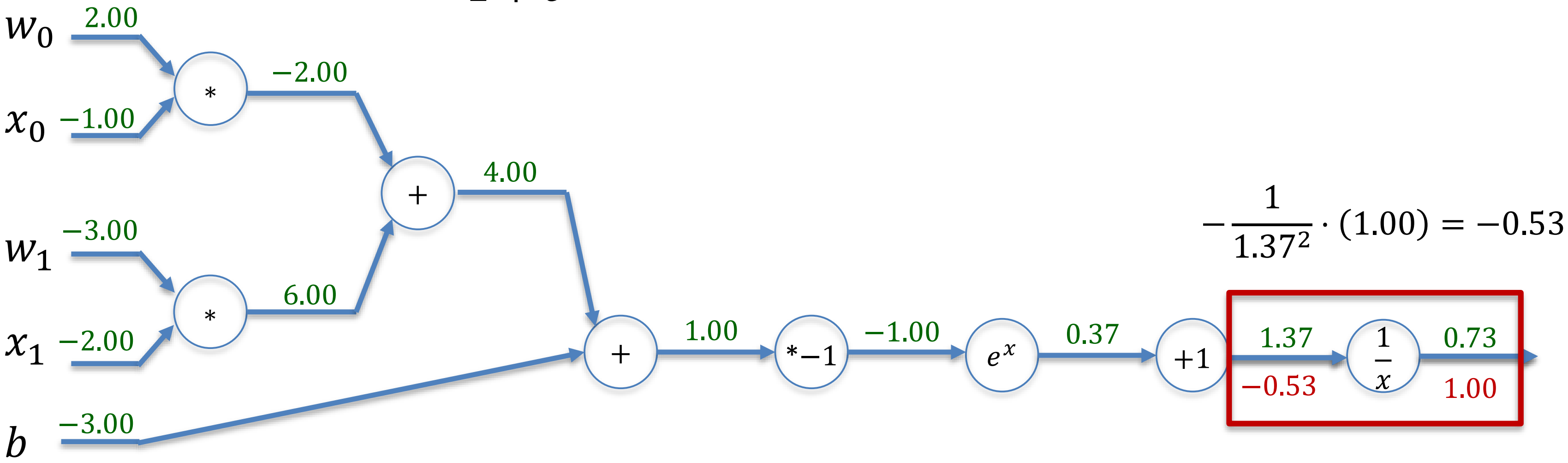
$$f_a(x) = ax \quad -> \quad \frac{\partial f_a}{\partial x} = a$$

$$f(x) = \frac{1}{x} \quad -> \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f_c(x) = c + x \quad -> \quad \frac{\partial f_c}{\partial x} = 1$$

Backprop

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



$$f(x) = e^x \quad -> \quad \frac{\partial f}{\partial x} = e^x$$

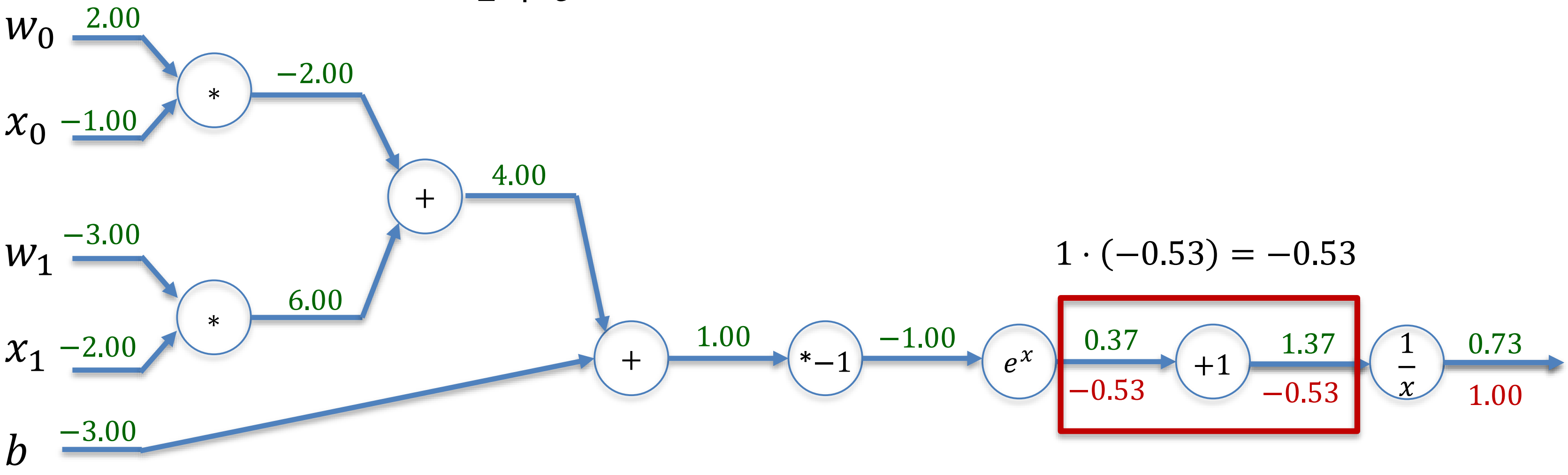
$$f_a(x) = ax \quad -> \quad \frac{\partial f_a}{\partial x} = a$$

$$\boxed{f(x) = \frac{1}{x} \quad -> \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}}$$

$$f_c(x) = c + x \quad -> \quad \frac{\partial f_c}{\partial x} = 1$$

Backprop

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



$f(x) = e^x \quad \text{->} \quad \frac{\partial f}{\partial x} = e^x$

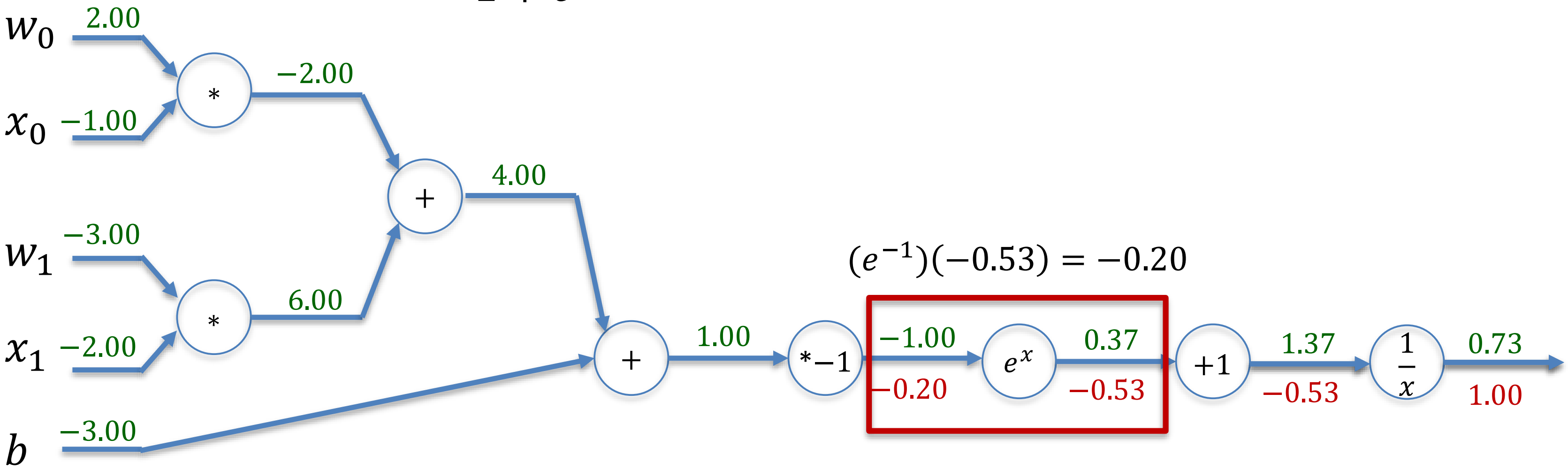
$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$

$f(x) = \frac{1}{x} \quad \text{->} \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$

$\boxed{f_c(x) = c + x \quad \text{->} \quad \frac{\partial f_c}{\partial x} = 1}$

Backprop

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



$$\boxed{f(x) = e^x \quad \text{->} \quad \frac{\partial f}{\partial x} = e^x}$$

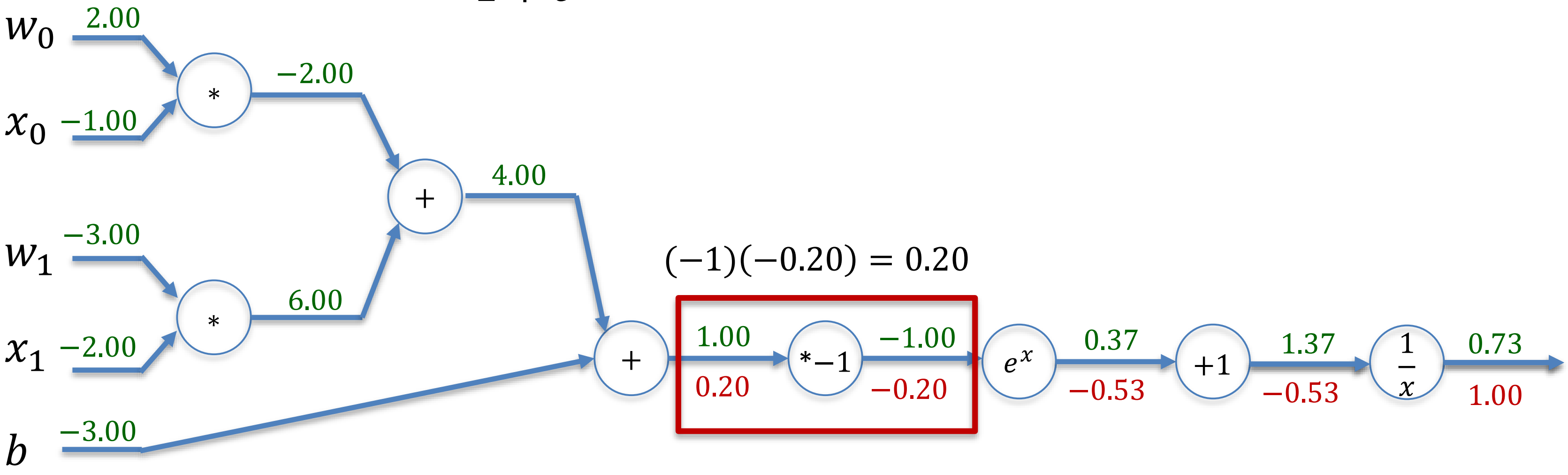
$$f(x) = \frac{1}{x} \quad \text{->} \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

$$f_c(x) = c + x \quad \text{->} \quad \frac{\partial f_c}{\partial x} = 1$$

Backprop

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



$$f(x) = e^x \quad \text{->} \quad \frac{\partial f}{\partial x} = e^x$$

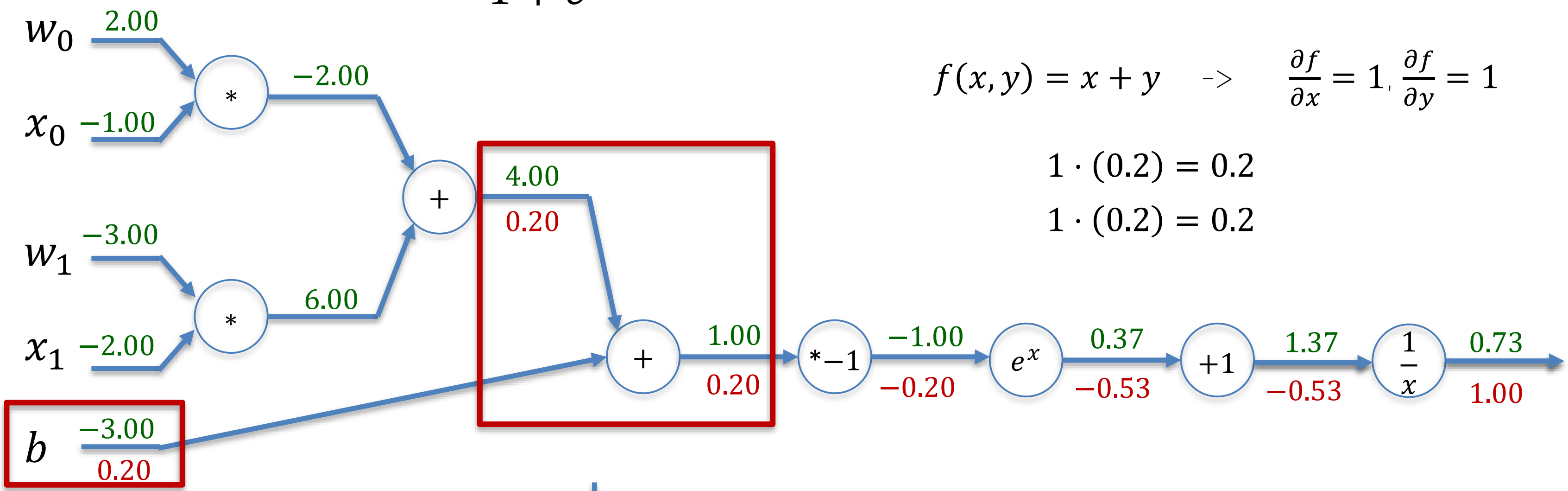
$$\boxed{f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a}$$

$$f(x) = \frac{1}{x} \quad \text{->} \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f_c(x) = c + x \quad \text{->} \quad \frac{\partial f_c}{\partial x} = 1$$

Backprop

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



$f(x, y) = x + y$ -> $\frac{\partial f}{\partial x} = 1, \frac{\partial f}{\partial y} = 1$

$1 \cdot (0.2) = 0.2$

$1 \cdot (0.2) = 0.2$

$f(x) = e^x$ -> $\frac{\partial f}{\partial x} = e^x$

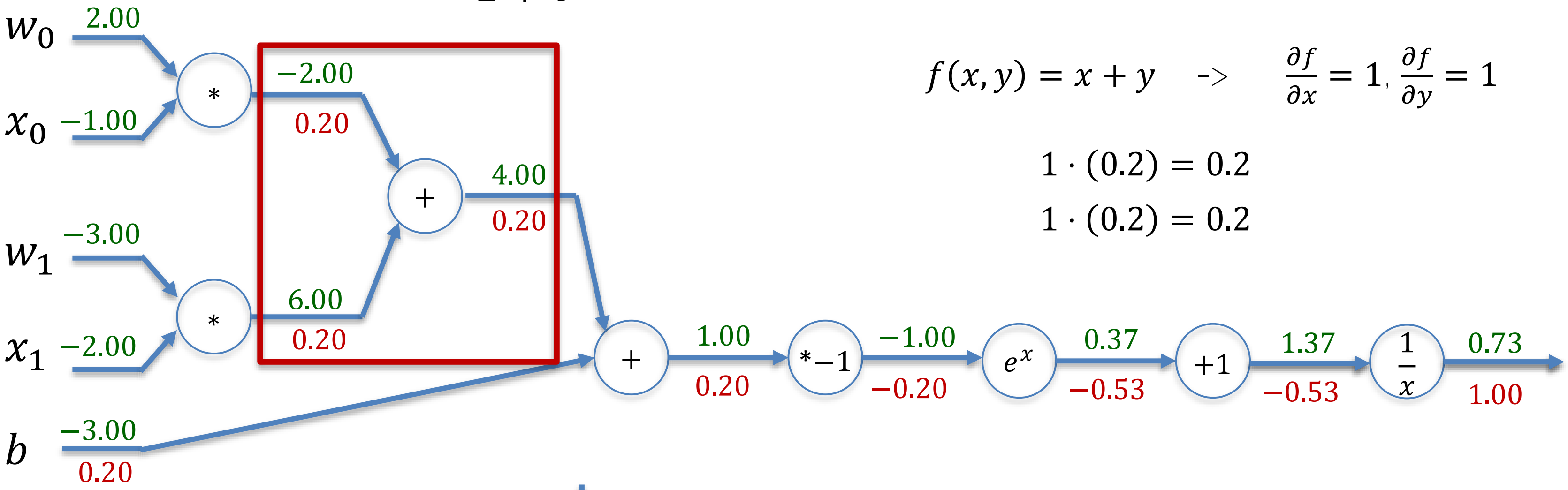
$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

$f(x) = \frac{1}{x}$ -> $\frac{\partial f}{\partial x} = -\frac{1}{x^2}$

$f_c(x) = c + x$ -> $\frac{\partial f_c}{\partial x} = 1$

Backprop

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



$f(x, y) = x + y$ -> $\frac{\partial f}{\partial x} = 1$, $\frac{\partial f}{\partial y} = 1$

$1 \cdot (0.2) = 0.2$

$1 \cdot (0.2) = 0.2$

$f(x) = e^x$ -> $\frac{\partial f}{\partial x} = e^x$

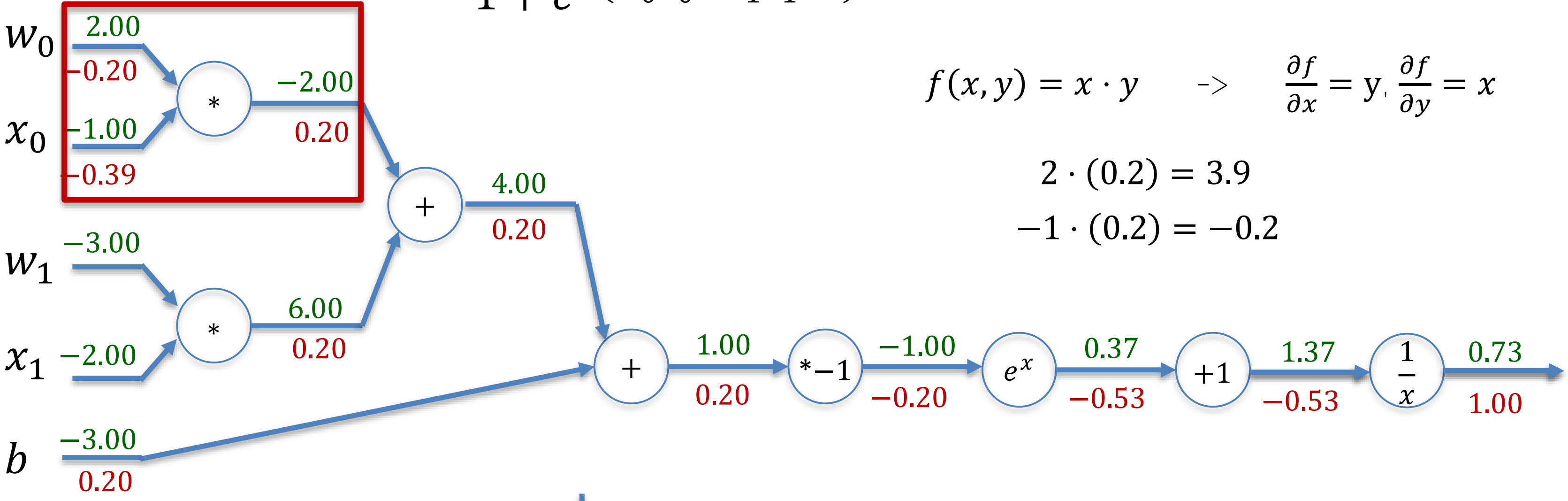
$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

$f(x) = \frac{1}{x}$ -> $\frac{\partial f}{\partial x} = -\frac{1}{x^2}$

$f_c(x) = c + x$ -> $\frac{\partial f_c}{\partial x} = 1$

Backprop

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



$f(x, y) = x \cdot y$ -> $\frac{\partial f}{\partial x} = y, \frac{\partial f}{\partial y} = x$

$$2 \cdot (0.2) = 3.9$$

$$-1 \cdot (0.2) = -0.2$$

$f(x) = e^x$ -> $\frac{\partial f}{\partial x} = e^x$

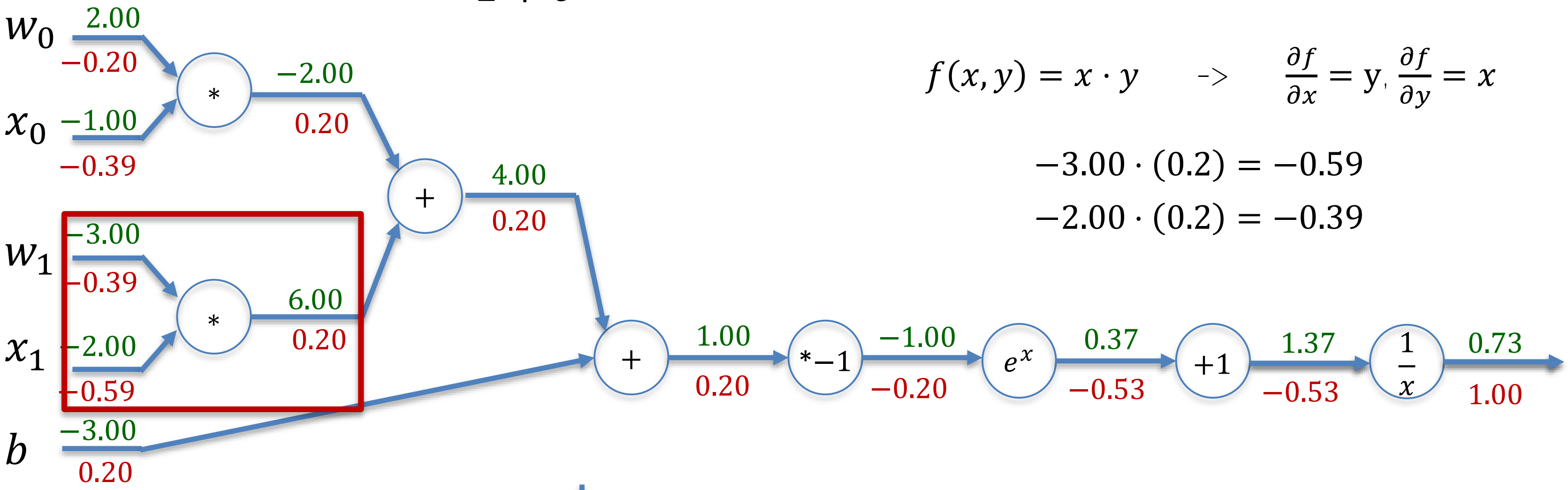
$f(x) = \frac{1}{x}$ -> $\frac{\partial f}{\partial x} = -\frac{1}{x^2}$

$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

$f_c(x) = c + x$ -> $\frac{\partial f_c}{\partial x} = 1$

Backprop

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



$f(x, y) = x \cdot y \quad \text{->} \quad \frac{\partial f}{\partial x} = y, \ \frac{\partial f}{\partial y} = x$

$-3.00 \cdot (0.2) = -0.59$

$-2.00 \cdot (0.2) = -0.39$

$f(x) = e^x \quad \text{->} \quad \frac{\partial f}{\partial x} = e^x$

$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$

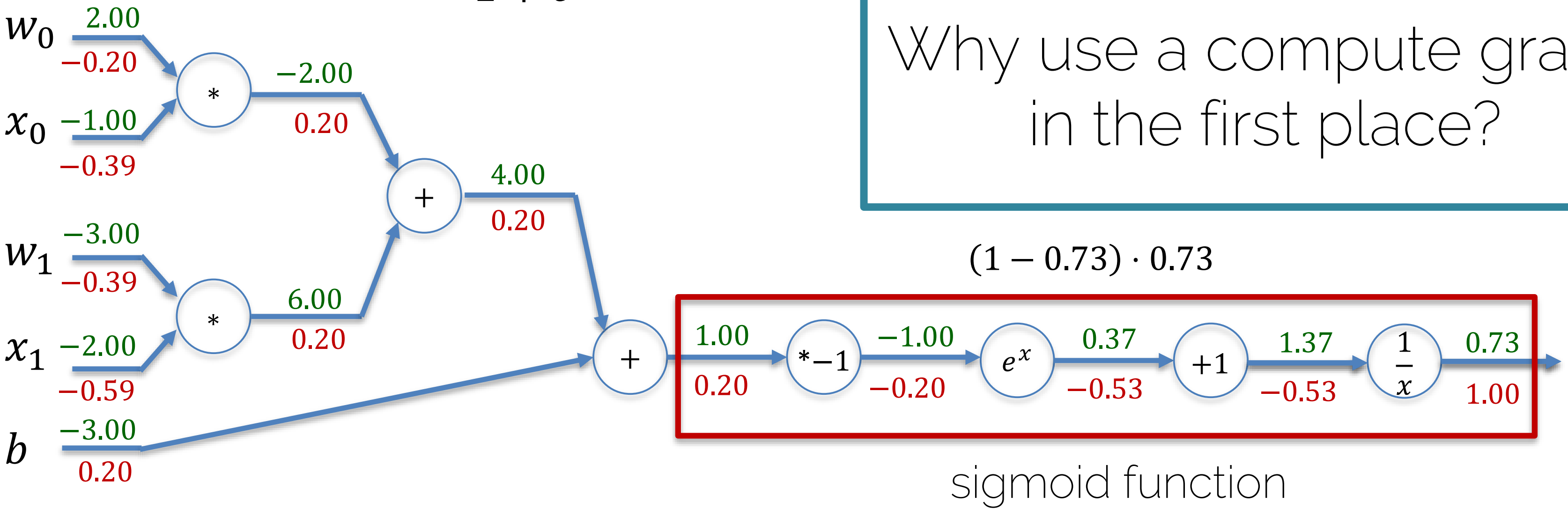
$f(x) = \frac{1}{x} \quad \text{->} \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$

$f_c(x) = c + x \quad \text{->} \quad \frac{\partial f_c}{\partial x} = 1$

Backprop

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

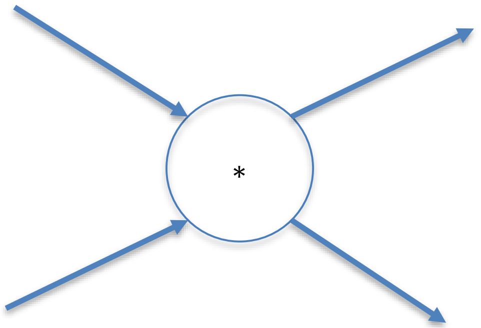
Why use a compute graph in the first place?



$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \text{->} \quad \frac{\partial \sigma(x)}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = \big(1 - \sigma(x)\big)\sigma(x)$$

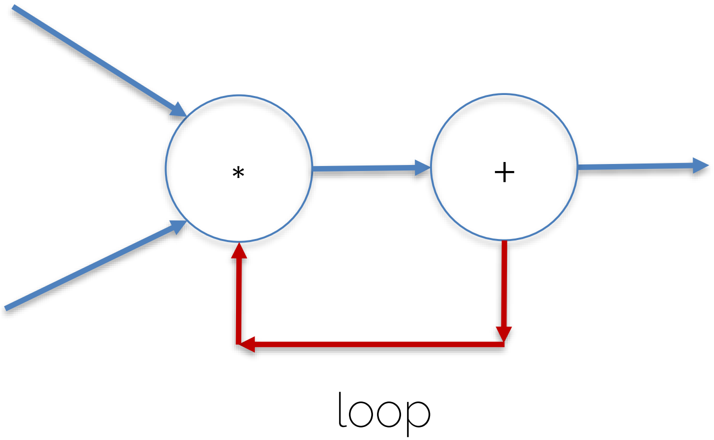
Backpropagation

What happens if there are multiple outputs in a compute node?



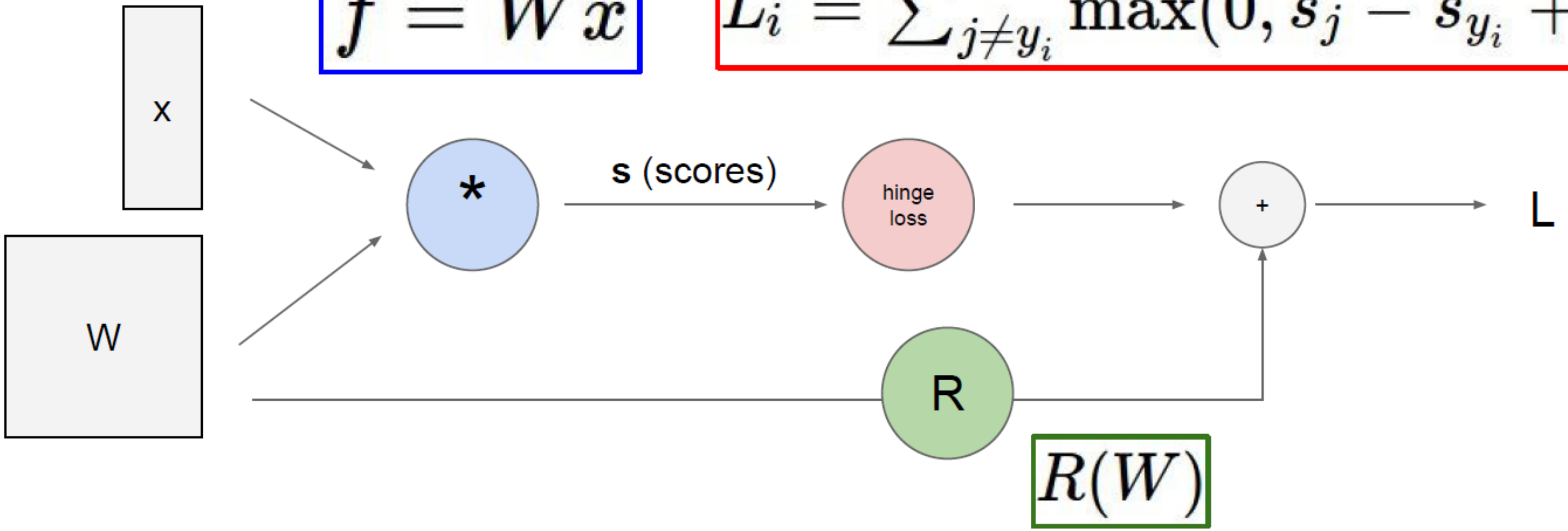
Backpropagation

What happens if there are loops in the graph?



Computational Graph

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



Combining nodes:
Linear activation node + hinge loss + regularization

Computational Graph: Logistic Regression

- Loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

- Cost function

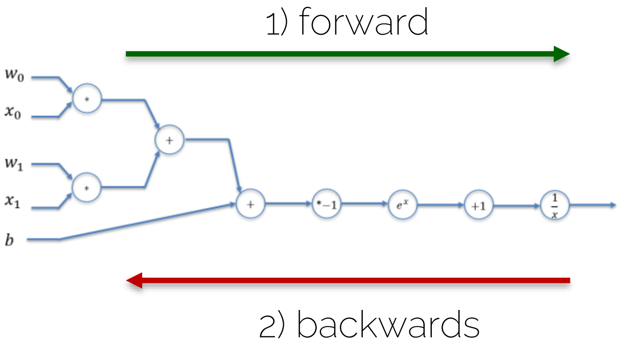
<u>*More Next Lecture!*</u>

$$C(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Minimization

$$\hat{y}_i = \sigma(\mathbf{x}_i \boldsymbol{\theta})$$

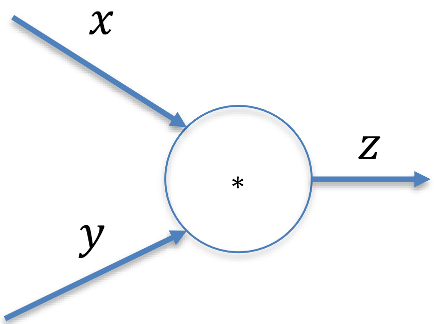
Implementation of Compute Graph




```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        #1. [pass inputs to input nodes]
        #2. forward traverse the computational graph
        for node in self.graph.nodes_topologically_sorted():
            node.forward()
            # forward intermediates / loss
        return loss # final node returns loss
    def backward():
        for node in self.graph.nodes_topologically_sorted_reverse():
            node.backward() #apply chainrule
            # backward intermediate derivatives
        return inputs_gradients
```

Implementation of Nodes

- Forward and backward pass of MulNode



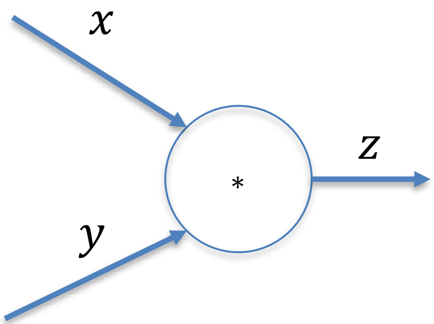
all values are scalars

```
class MulNode(object):
    def forward(x,y):
        z=x*y
        return z
    def backward(dz, x, y):
        dx = y*dz    # [dz/dx * dL/dz]
        dy = x*dz    # [dz/dy * dL/dz]
        return [dx, dy]
```

Issue?

Implementation of Nodes

- Forward and backward pass of MulNode



all values are scalars

```python
class MulNode(object):
    def forward(x,y):
        z = x*y
        self.x = x
        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]
```

Cache results of forward pass
-> faster runtime for backward pass

Torch: Layers (GitHub)

LayerNormalization.lua	Adding Layer Normalization	2 months ago
LeakyReLU.lua	Add THNN conversion of {ELU, LeakyReLU, LogSigmoid, LogSoftMax, Looku...	a year ago
Linear.lua	Fix shared function override for specific modules	4 months ago
Log.lua	add nn.Inc & nn.Scale	a year ago
LogSigmoid.lua	lazy init	a year ago
LogSoftMax.lua	Add THNN conversion of {ELU, LeakyReLU, LogSigmoid, LogSoftMax, Looku...	a year ago
LookupTable.lua	Fix shared function override for specific modules	4 months ago
MM.lua		
MSECriterion.lua		
MV.lua		
MapTable.lua		
MarginCriterion.lua		
MarginRankingCriterion.lu		
MaskedSelect.lua		
Max.lua		
Maxout.lua		
Mean.lua		
Min.lua		
MixtureTable.lua		
Module.lua		
Mul.lua		
MulConstant.lua		
MultiCriterion.lua		

Reshape.lua	Better __tostring__ and cleans formatting	a year ago
Select.lua	Adds negative dim arguments	11 months ago
SelectTable.lua	allow SelectTable to accept input that contains tables of things that...	2 months ago
Sequential.lua	Improve error handling	a year ago
Sigmoid.lua	Add THNN conversion of {RReLU, Sigmoid, SmoothL1Criterion,SoftMax, So...	a year ago
SmoothL1Criterion.lua	Add THNN conversion of {RReLU, Sigmoid, SmoothL1Criterion,SoftMax, So...	a year ago
SoftMarginCriterion.lua	SoftMarginCriterion	a year ago
SoftMax.lua	Add THNN conversion of {RReLU, Sigmoid, SmoothL1Criterion,SoftMax, So...	a year ago
SoftMin.lua	nn.clearState	a year ago
SoftPlus.lua	Add THNN conversion of {RReLU, Sigmoid, SmoothL1Criterion,SoftMax, So...	a year ago
SoftShrink.lua	Add THNN conversion of {oftShrink, Sqrt, Square, Tanh, Threshold}	a year ago
SoftSign.lua	nn.clearState	a year ago
SparseJacobian.lua	Fix various unused variables in nn	3 years ago
SparseLinear.lua	Fixing sparse linear race condition	a year ago
SpatialAdaptiveAveragePooling.lua	Add SpatialAdaptiveAveragePooling.	4 months ago
SpatialAdaptiveMaxPooling.lua	Indices for nn.	7 months ago
SpatialAutoCropMSECriterion.lua	fix local / global var leaks	4 months ago

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:set(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:set(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
```

Init()

$$f(x) = aX$$

Forward()

Backward()

Caffee: Layers (GitHub)

absval_layer.cpp	dismantle layer headers	2 years ago
absval_layer.cu	dismantle layer headers	2 years ago
accuracy_layer.cpp		
argmax_layer.cpp		
base_conv_layer.cpp		
base_data_layer.cpp		
base_data_layer.cu		
batch_norm_layer.cpp		
batch_norm_layer.cu		
batch_reindex_layer.cpp		
batch_reindex_layer.cu		
bnll_layer.cpp		
bnll_layer.cu		
concat_layer.cpp		
concat_layer.cu		
contrastive_loss_layer.cpp		
contrastive_loss_layer.cu		
conv_layer.cpp		

concat_layer.cu	dismantle layer headers	2 years ago
contrastive_loss_layer.cpp	dismantle layer headers	2 years ago
contrastive_loss_layer.cu	dismantle layer headers	2 years ago
conv_layer.cpp		
conv_layer.cu		
cudnn_conv_layer.cpp		
cudnn_conv_layer.cu		
cudnn_lcn_layer.cpp		
cudnn_lcn_layer.cu		
cudnn_lrn_layer.cpp		
cudnn_lrn_layer.cu		
cudnn_pooling_layer.cpp		
cudnn_pooling_layer.cu		
cudnn_relu_layer.cpp		
cudnn_relu_layer.cu		
cudnn_sigmoid_layer.cpp		
cudnn_sigmoid_layer.cu		
cudnn_softmax_layer.cpp		
cudnn_softmax_layer.cu		
cudnn_tanh_layer.cpp		

pooling_layer.cpp	dismantle layer headers	2 years ago
pooling_layer.cu	dismantle layer headers	2 years ago
power_layer.cpp	dismantle layer headers	2 years ago
power_layer.cu	dismantle layer headers	2 years ago
prelu_layer.cpp	dismantle layer headers	2 years ago
prelu_layer.cu	dismantle layer headers	2 years ago
reduction_layer.cpp	dismantle layer headers	2 years ago
reduction_layer.cu	dismantle layer headers	2 years ago
relu_layer.cpp	dismantle layer headers	2 years ago
relu_layer.cu	dismantle layer headers	2 years ago
reshape_layer.cpp	dismantle layer headers	2 years ago
sigmoid_cross_entropy_loss_layer.cpp	dismantle layer headers	2 years ago
sigmoid_cross_entropy_loss_layer.cu	dismantle layer headers	2 years ago
$\boxed{\text{sigmoid\_layer.cpp}}$	dismantle layer headers	2 years ago
sigmoid_layer.cu	dismantle layer headers	2 years ago
silence_layer.cpp	dismantle layer headers	2 years ago
silence_layer.cu	dismantle layer headers	2 years ago
slice_layer.cpp	dismantle layer headers	2 years ago

Caffe: Sigmoid_Layer

```cpp
#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);
    }
  }
}
```

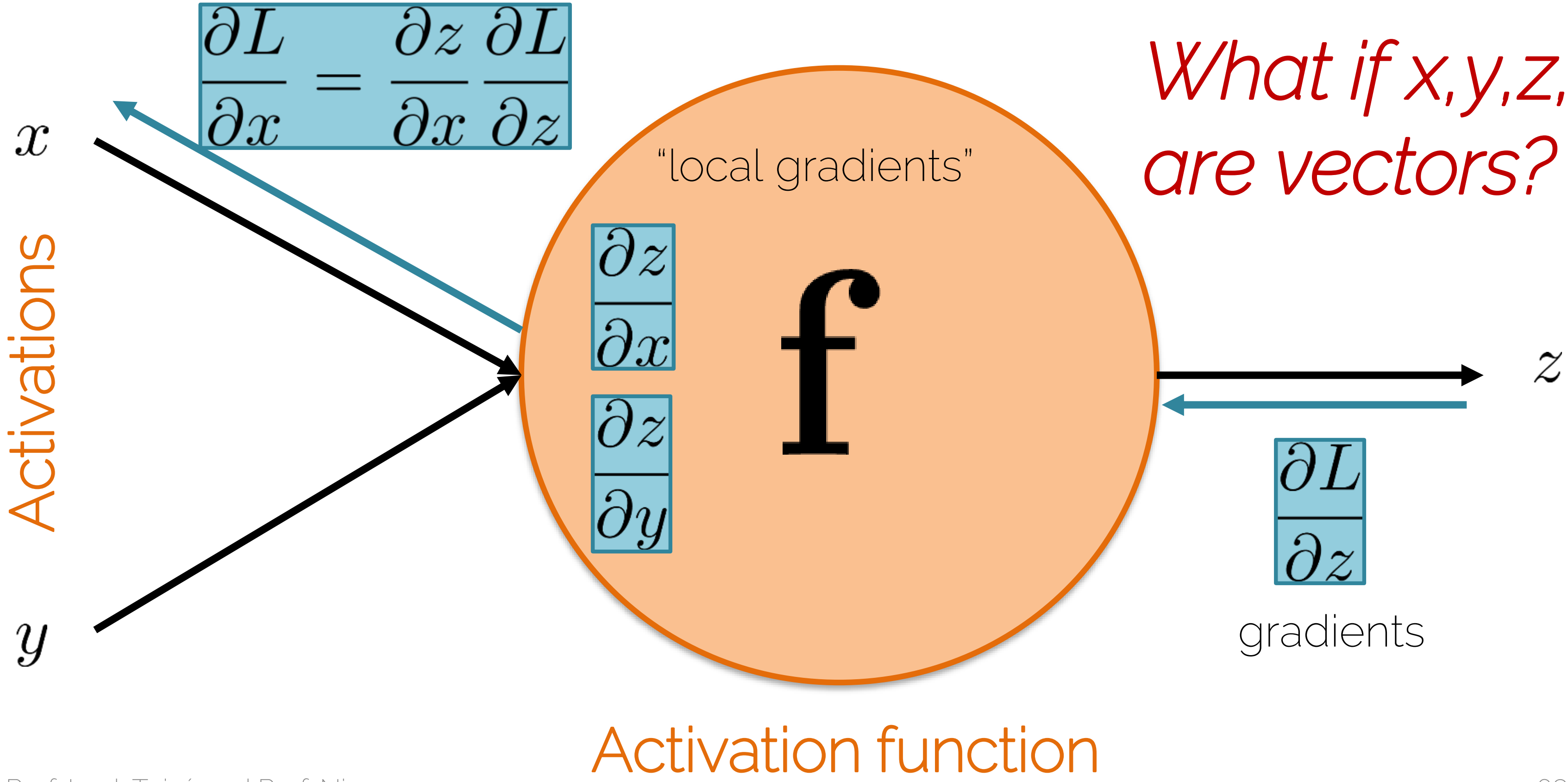
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Forward()

Backward()

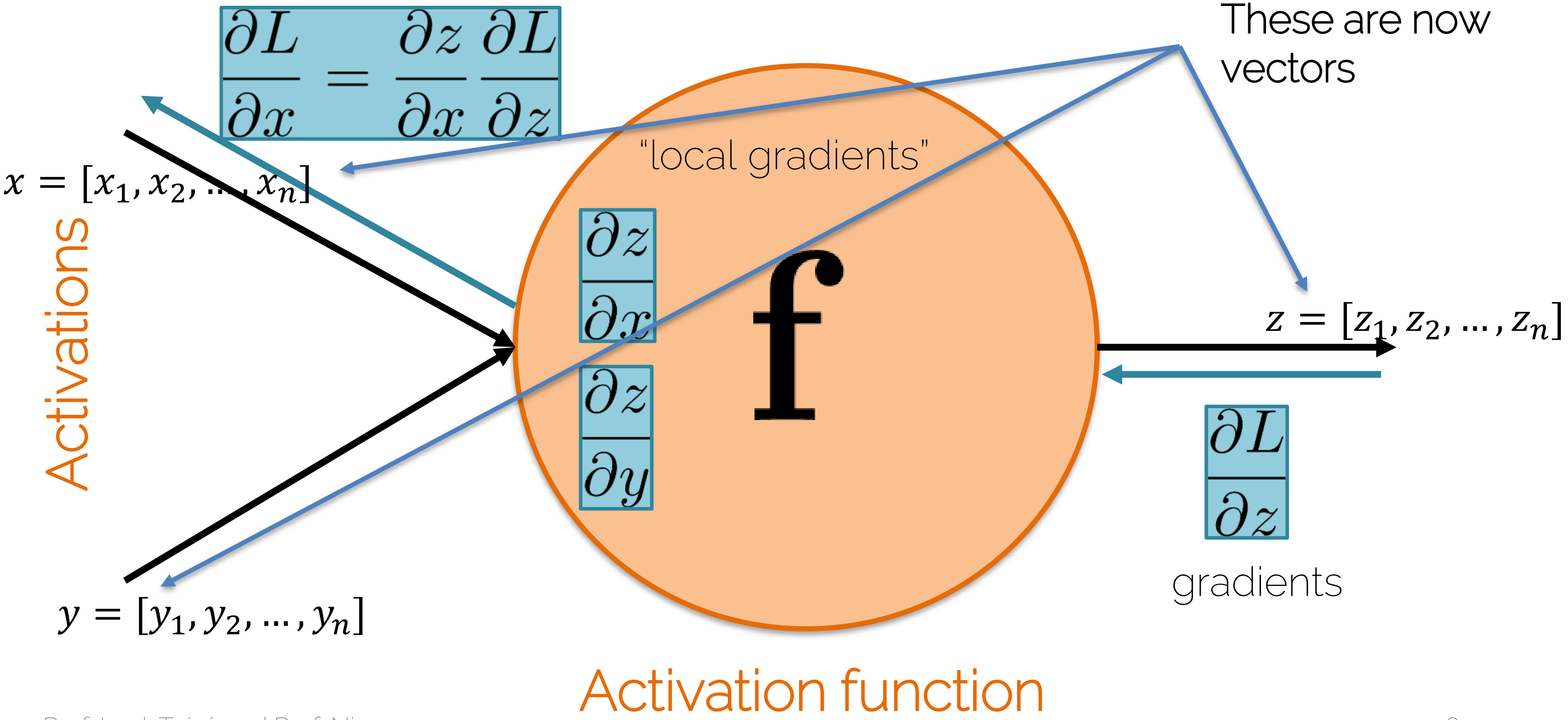
$$\sigma'(x) = \big(1 - \sigma(x)\big)\sigma(x)$$

Vectorized Operations

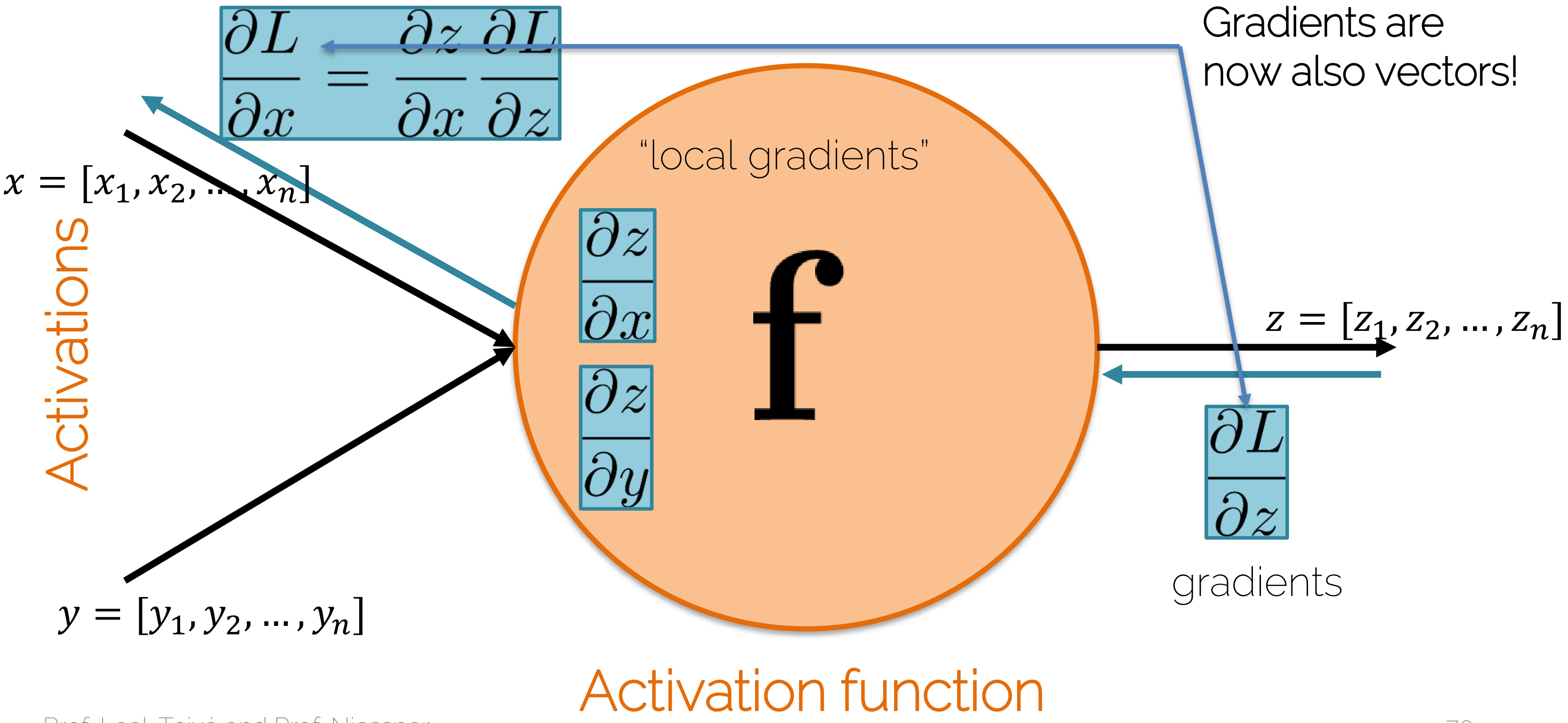


What if x,y,z, are vectors?

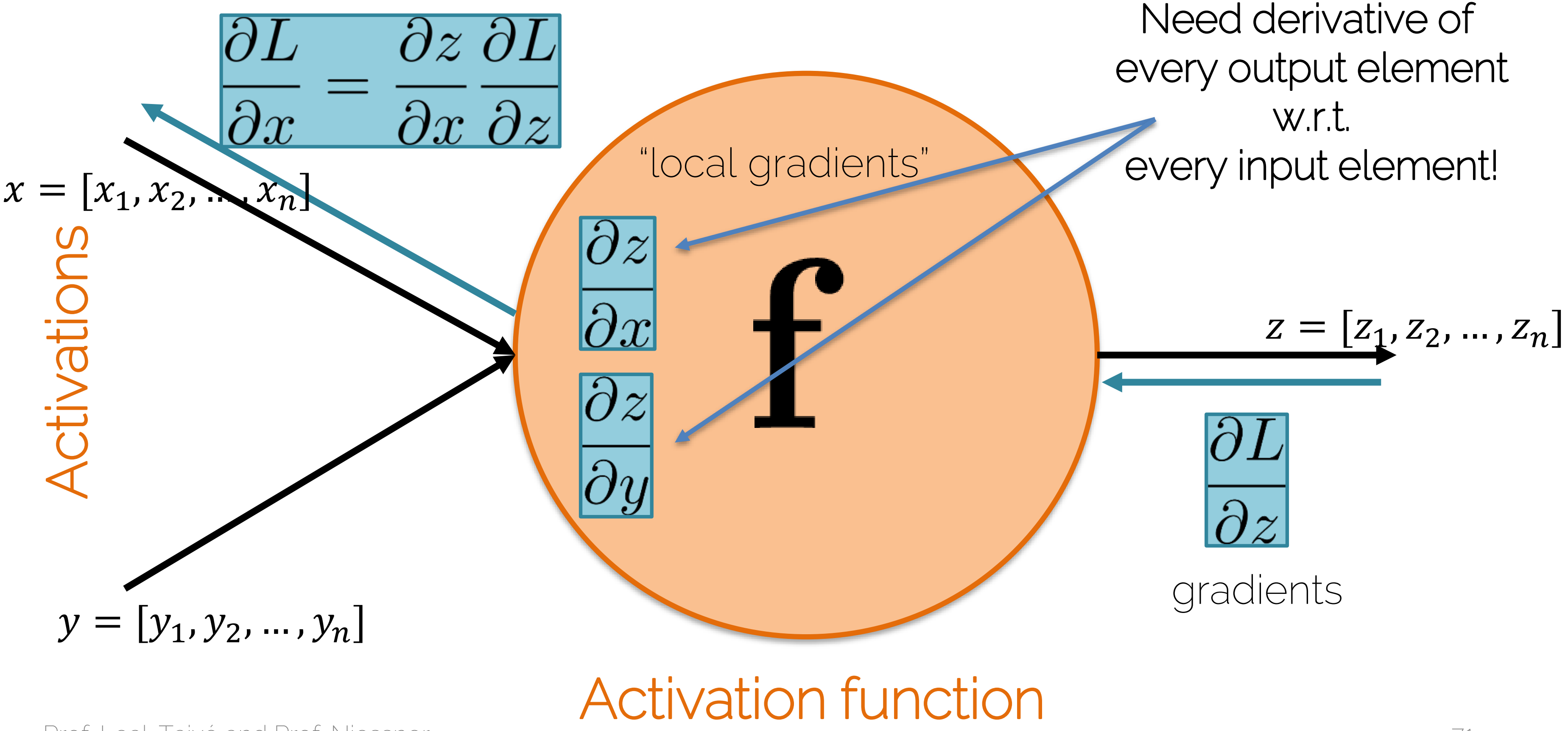
Vectorized Operations



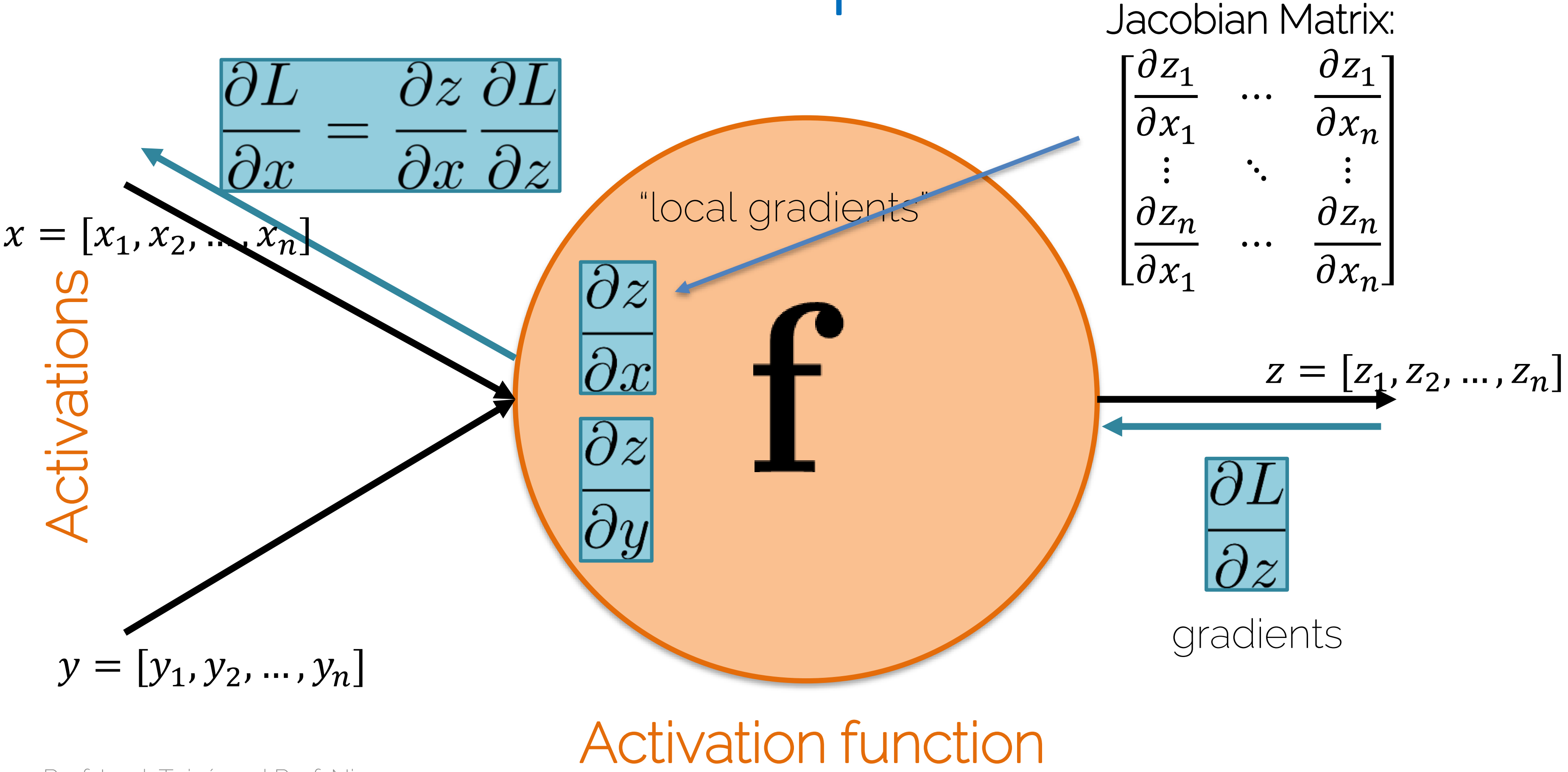
Vectorized Operations



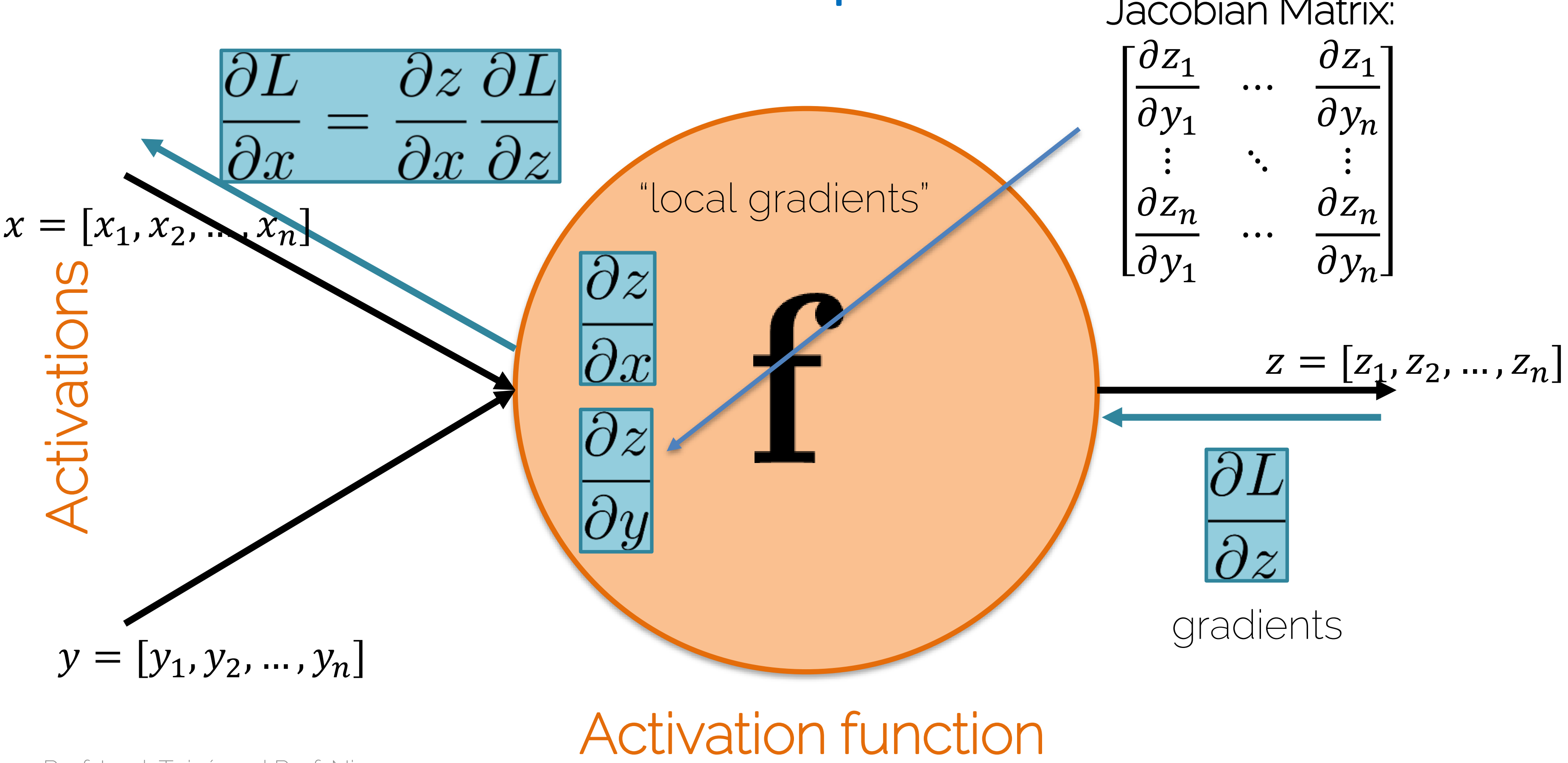
Vectorized Operations



Vectorized Operations



Vectorized Operations



Vectorized Operations

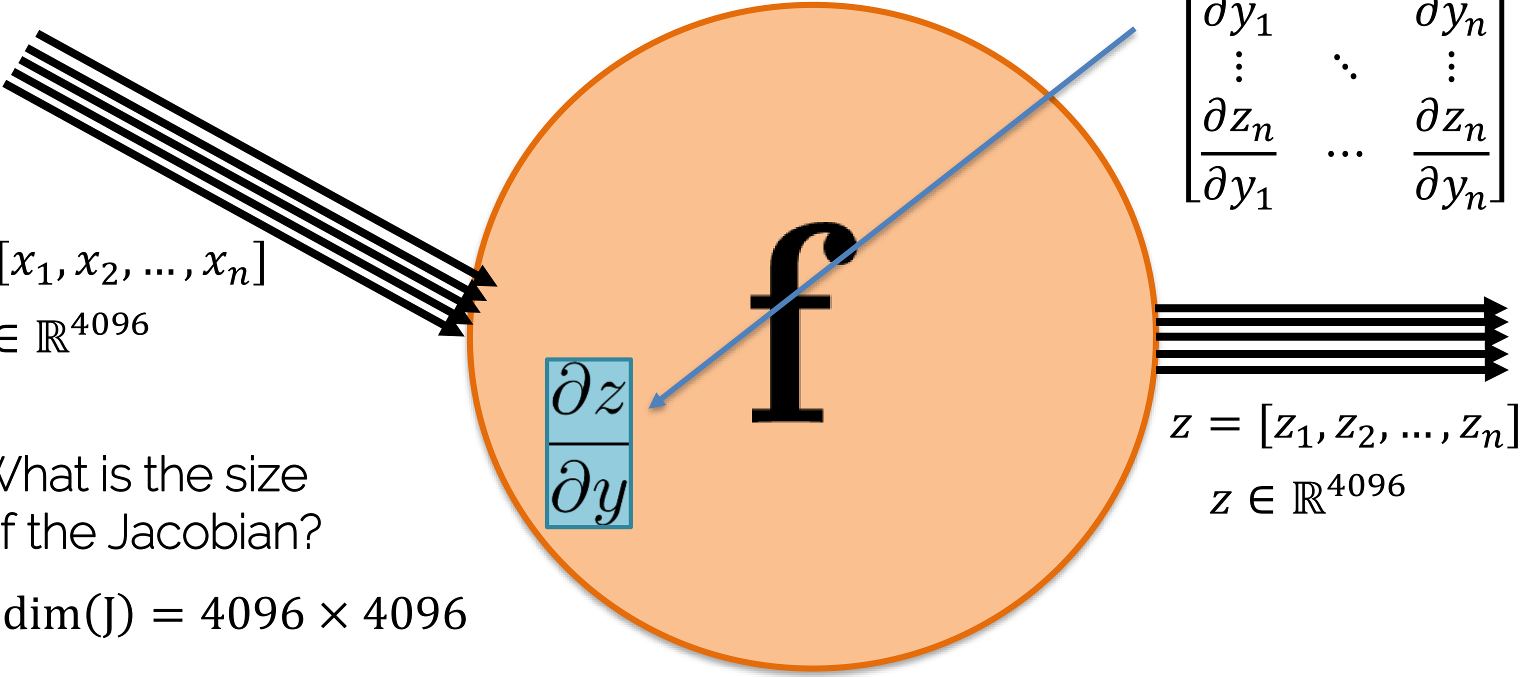
Assuming input and output $\in \mathbb{R}^{4096}$

Jacobian Matrix:

$$\begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \cdots & \frac{\partial z_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \cdots & \frac{\partial z_n}{\partial y_n} \end{bmatrix}$$

$x = [x_1, x_2, \ldots, x_n]$

$x \in \mathbb{R}^{4096}$



$z = [z_1, z_2, \ldots, z_n]$

$z \in \mathbb{R}^{4096}$

What is the size of the Jacobian?

$$\dim(\mathrm{J}) = 4096 \times 4096$$

Vectorized Operations

Jacobian Matrix:

$$\begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \cdots & \frac{\partial z_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \cdots & \frac{\partial z_n}{\partial y_n} \end{bmatrix}$$

How efficient is that:

- $\dim(\mathrm{J}) = 4096 \times 4096 = 16.78\,\mathrm{mio}$
- Assuming floats (i.e., 4 bytes / elem)
- -> 64 MB

Typically, networks are run in batches:

- Assuming mini-batch size of 16
- -> $\dim(\mathrm{J}) = (16 \cdot 4096) \times (16 \cdot 4096) = 4295\ \mathrm{mio}$
- -> 16.384MB = 16GB

How to handle this?

Neural Networks

Neural Networks

- Are compute graphs
- Goal: for given train set, find optimal weights

- Optimization using gradient-based solvers
  – Many options (more in the next lectures)

- Gradients are computed via backpropagation
  – Nice because can easily modularize complex functions

Administrative Things

- This week: tutorial on April 26th
  - Python tutorial
  - Introduction to submission system
- Next Lecture on April 30th
  - Optimization and Regularization
  - More on neural networks ☺
- Next week: first exercise
  - May 3rd -> start of exercise 1

See you next week!