Lecture 3 recap

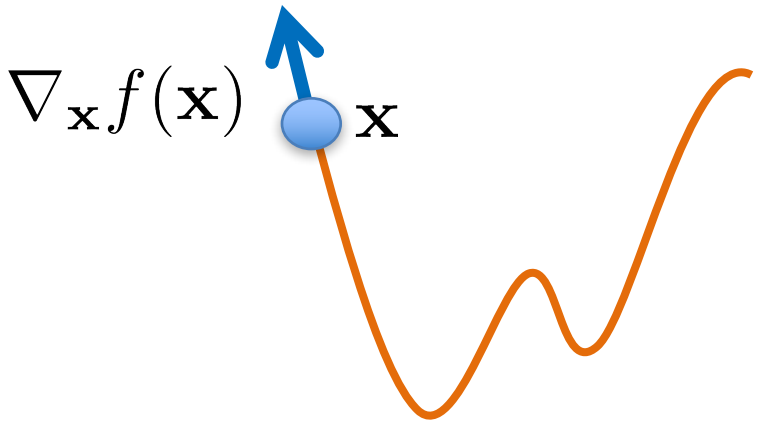
Gradient Descent

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

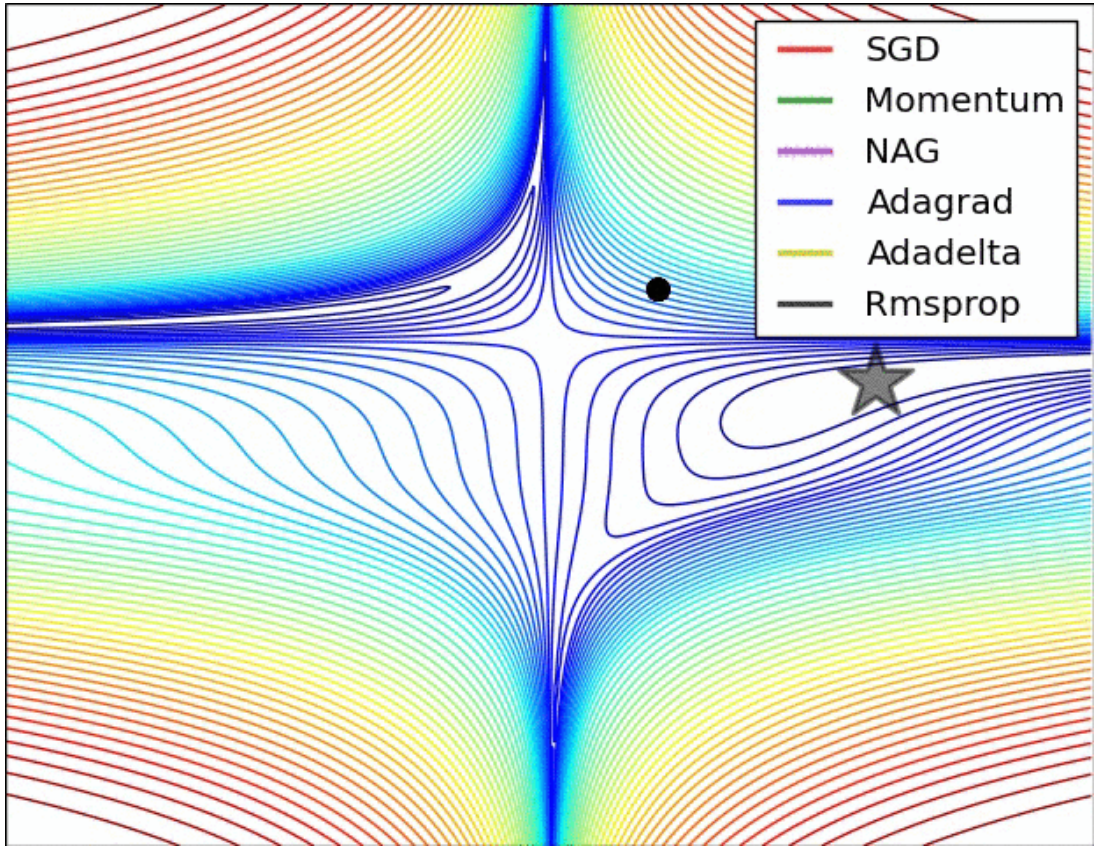
- Gradient steps in direction of negative gradient



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

Learning rate

Convergence



Numerical vs Analytical Gradient

Given the function $f(x) = \frac{1}{x}$

-> compute $f'(1.0)$

Numerical vs Analytical Gradient

Given the function $f(x) = \frac{1}{x}$

-> compute $f'(1.0)$

Numerical gradient:

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

for $h = 0.001$

$$f'(1.0) = \frac{f(1.001) - f(1.0)}{0.001} = -0.999$$

Numerical vs Analytical Gradient

Given the function $f(x) = \frac{1}{x}$

-> compute $f'(1.0)$

Numerical gradient:

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

for $h = 0.001$

$$f'(1.0) = \frac{f(1.001) - f(1.0)}{0.001} = -0.999$$

Analytical gradient gradient:

$$\frac{d}{dx}f(x) = -\frac{1}{x^2}$$

$$f'(1.0) = -\frac{1}{1.0^2} = -1$$

Numerical vs Analytical Gradient

Given the function $f(x) = \frac{1}{x}$

-> compute $f'(1.0)$

Numerical gradient:

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

for $h = 0.001$

$$f'(1.0) = \frac{f(1.001) - f(1.0)}{0.001} = -0.999$$

Slow ☹, approximate ☹, easy-to-write ☺

Analytical gradient gradient:

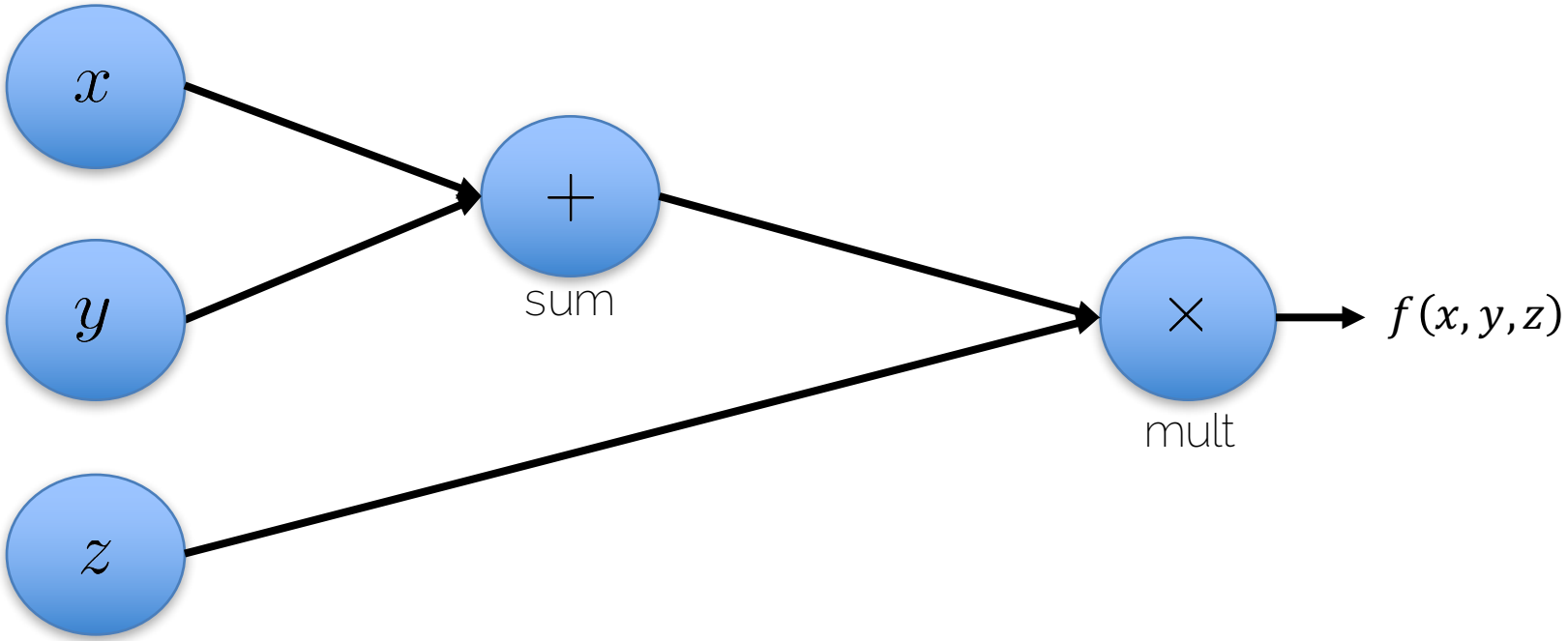
$$\frac{d}{dx}f(x) = -\frac{1}{x^2}$$

$$f'(1.0) = -\frac{1}{1.0^2} = -1$$

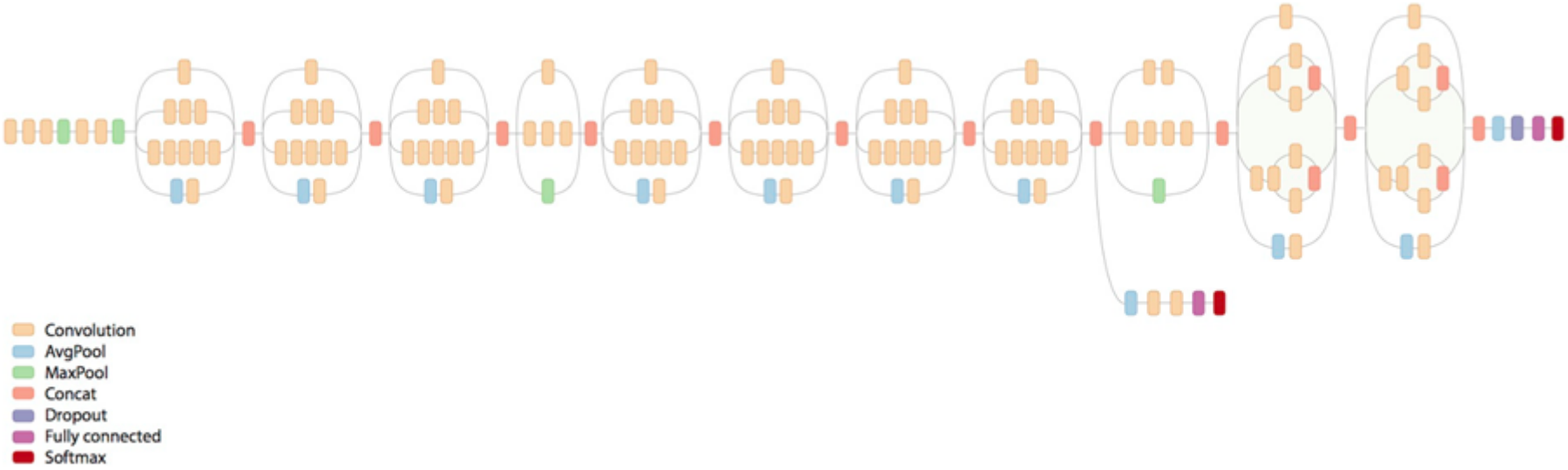
Fast ☺, exact* ☺, error-prone ☹

Computational Graphs

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



Computational Graphs



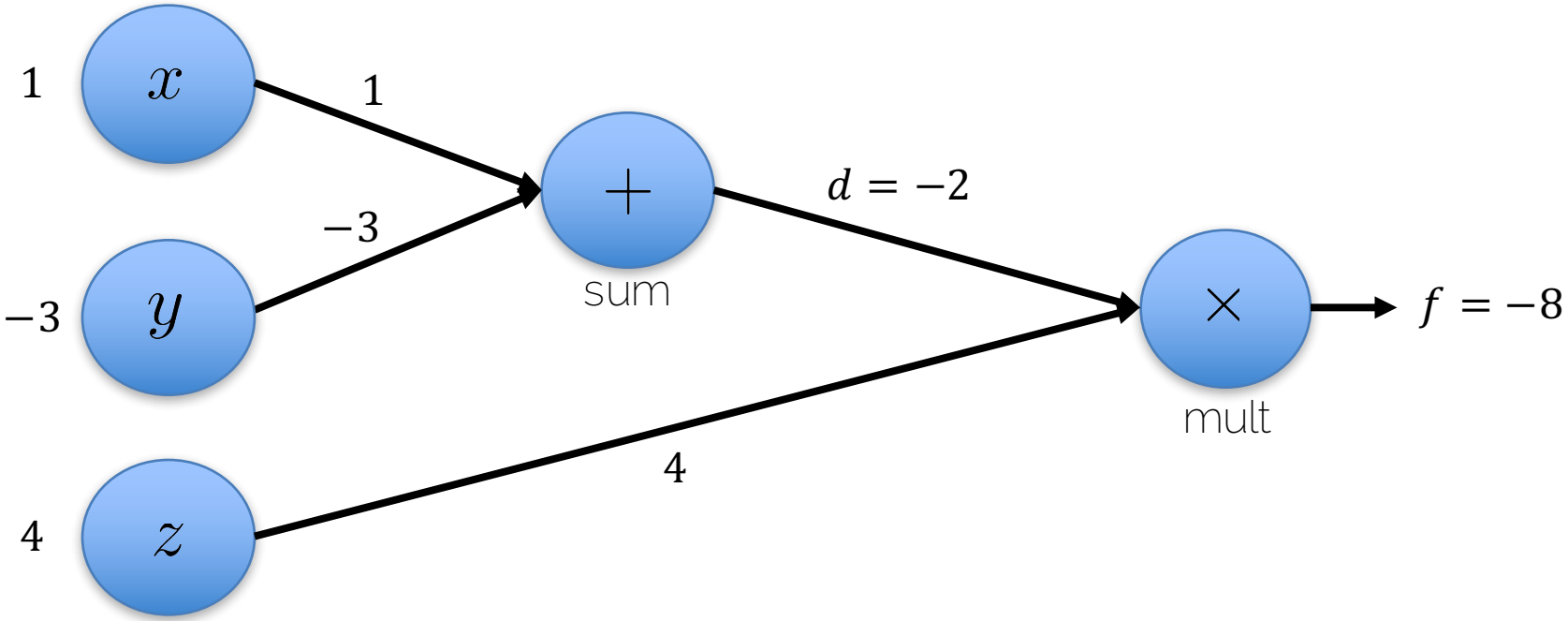
Another view of GoogLeNet's architecture.

Backprop: cont'd

Backprop: Forward Pass

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

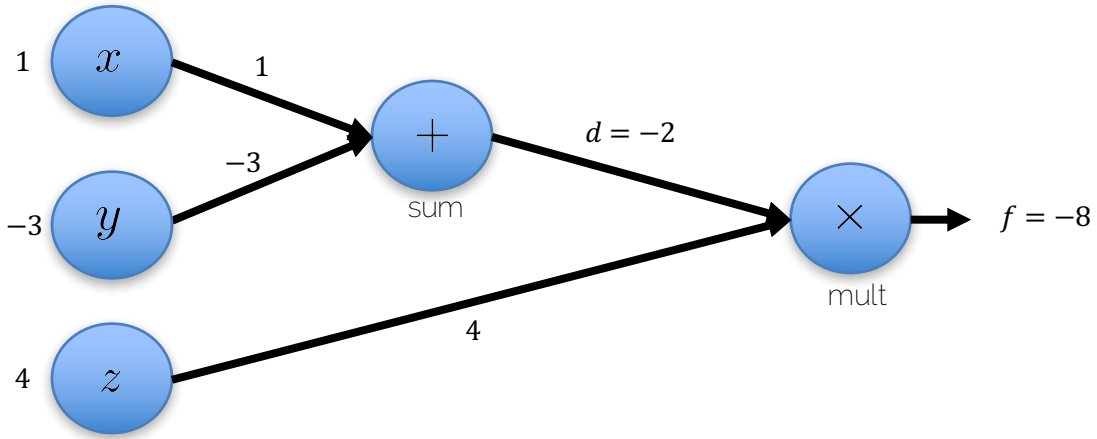
Initialization $x = 1, y = -3, z = 4$



Backprop: Backward Pass

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

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



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

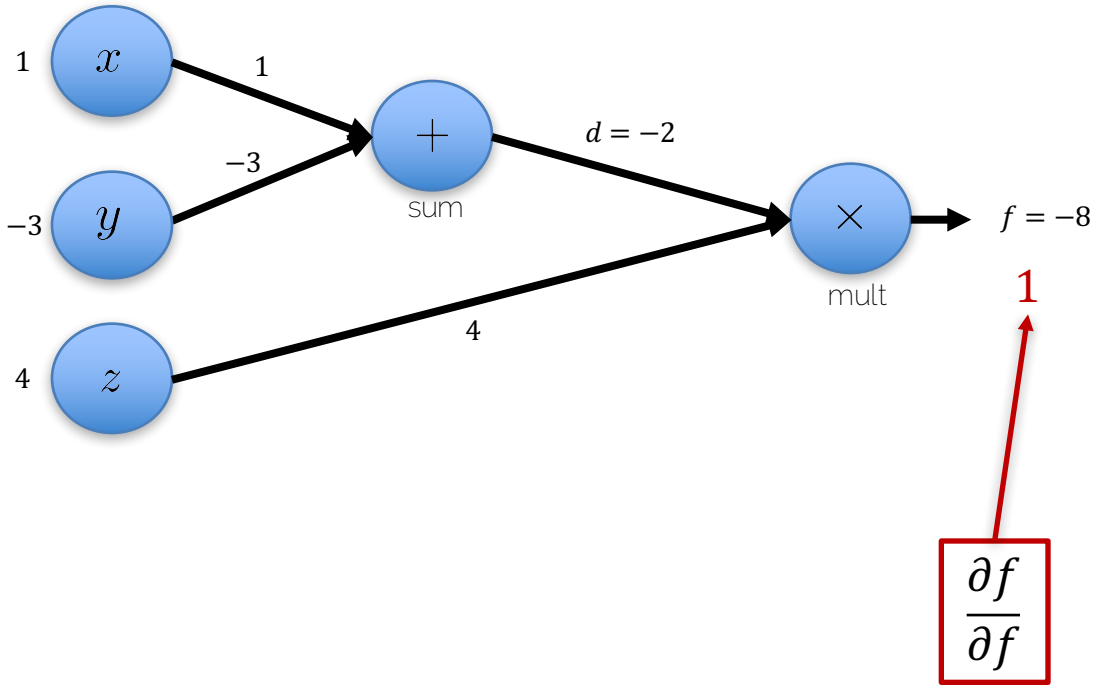
$$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$$

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



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

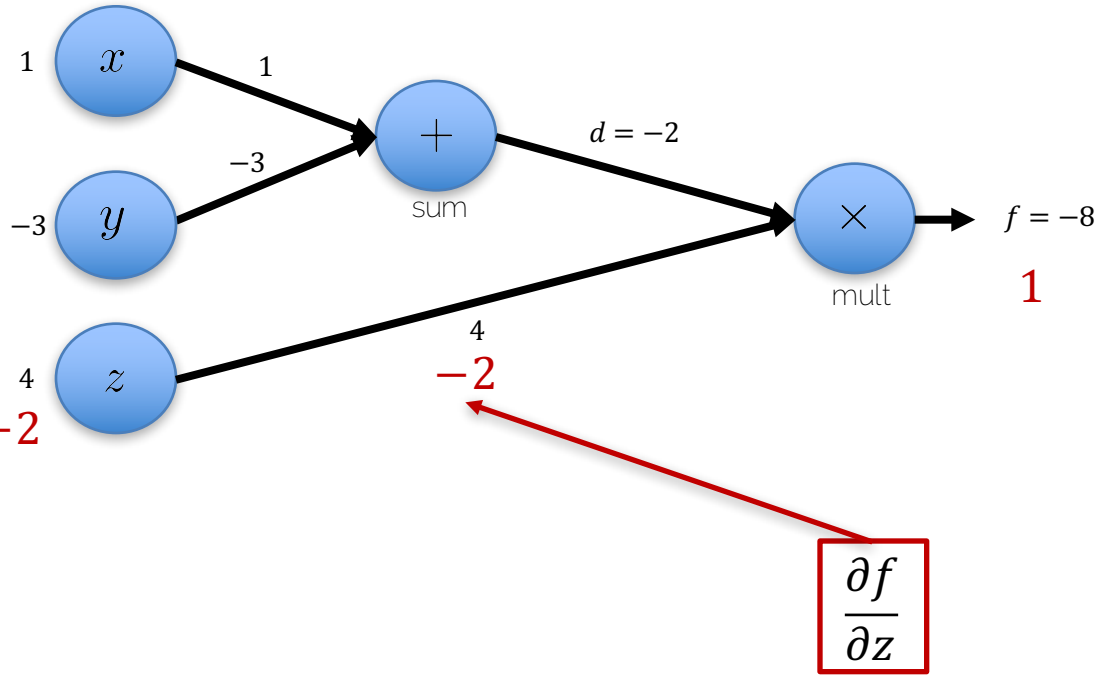
$$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$$

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



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

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

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

Backprop: Backward Pass

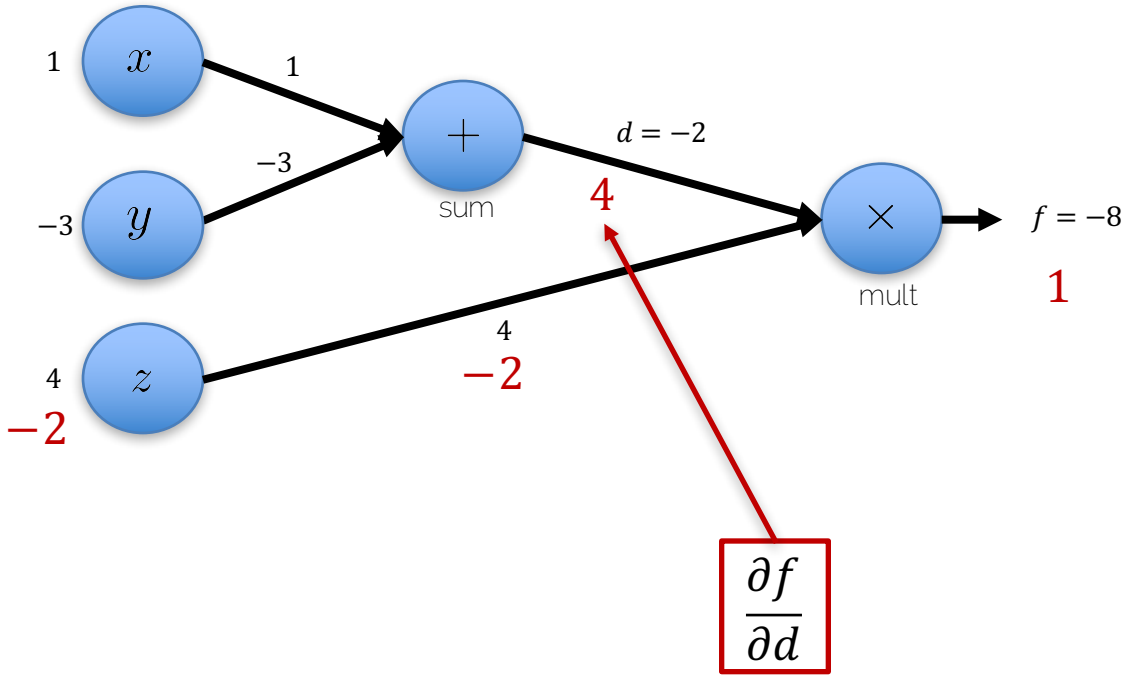
$f(x, y, z) = (x + y) \cdot z$

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

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

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

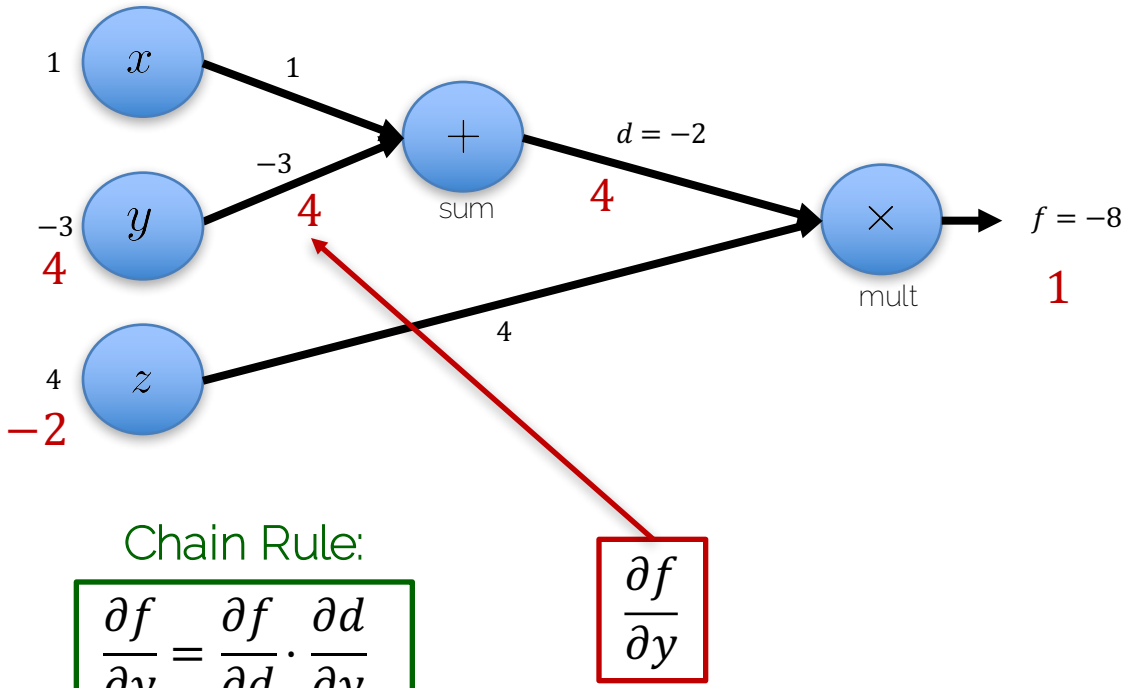
What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?



Backprop: Backward Pass

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

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



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

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

Chain Rule:

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

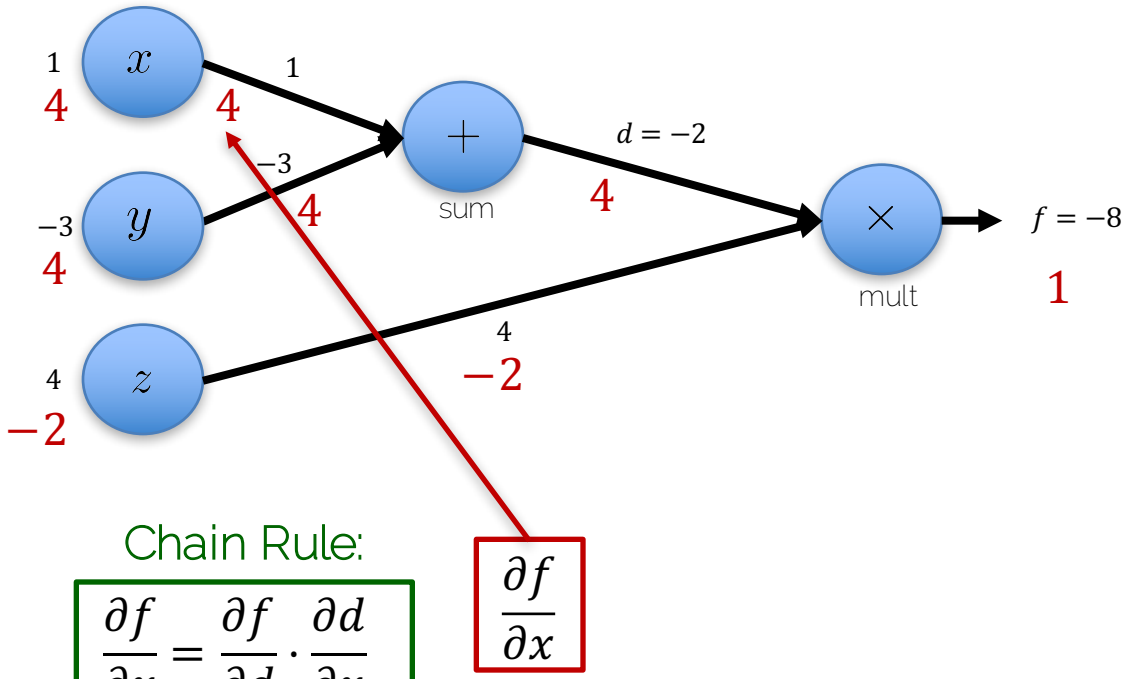
$$\rightarrow \frac{\partial f}{\partial y} = 4 \cdot 1 = 4$$

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



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

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

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

Chain Rule:

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

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

Backprop: Backward Pass

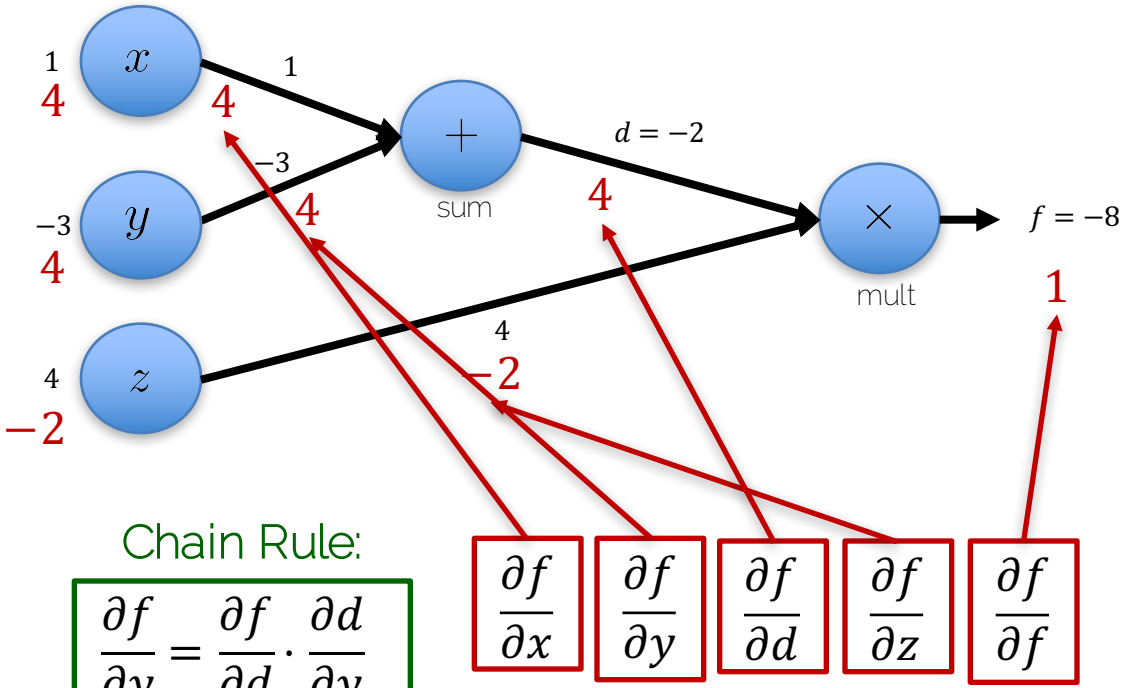
$$f(x, y, z) = (x + y) \cdot z$$

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

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

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

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

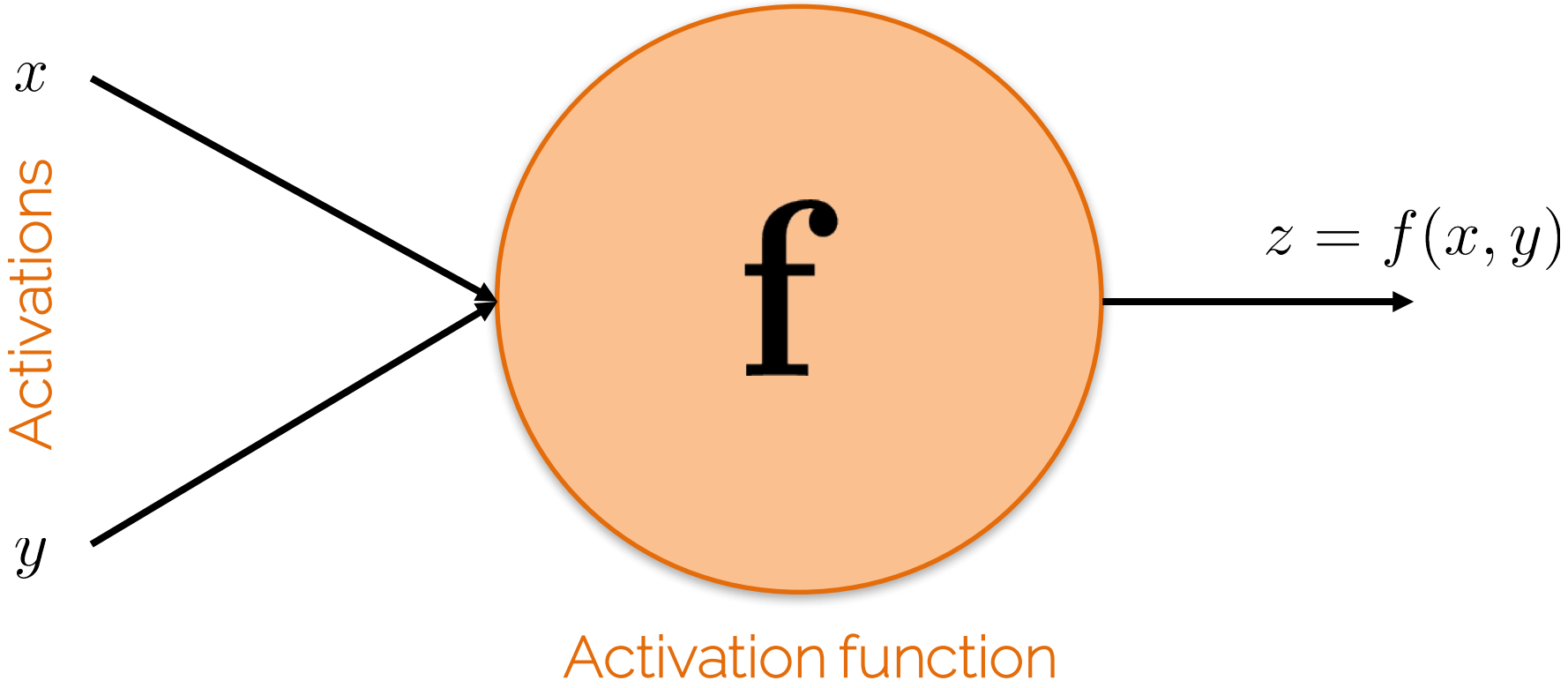


Chain Rule:

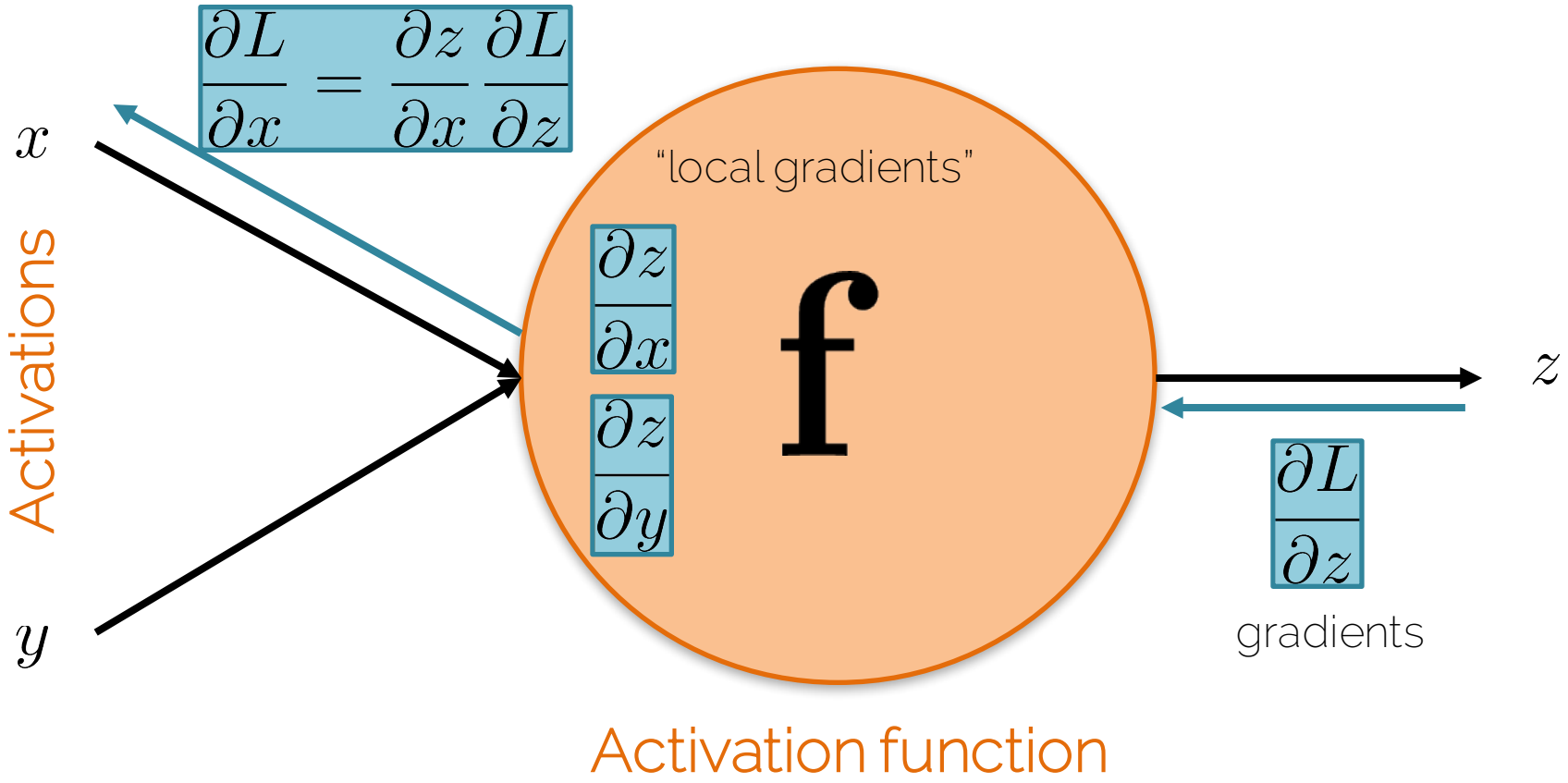
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial y}$$

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

The Flow of Gradients

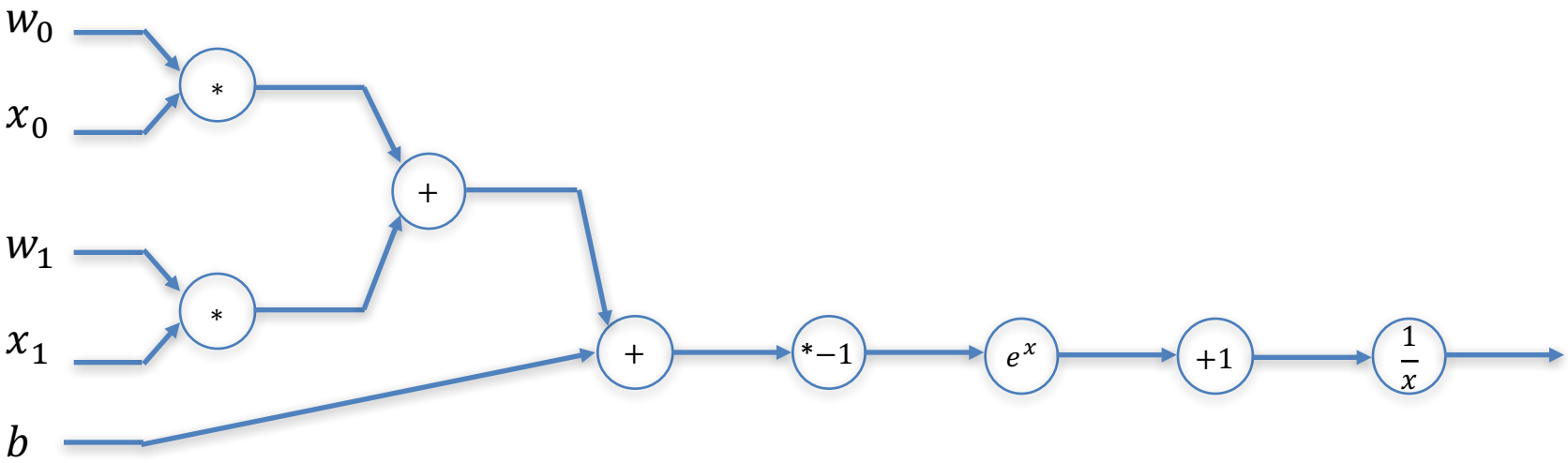


The Flow of Gradients



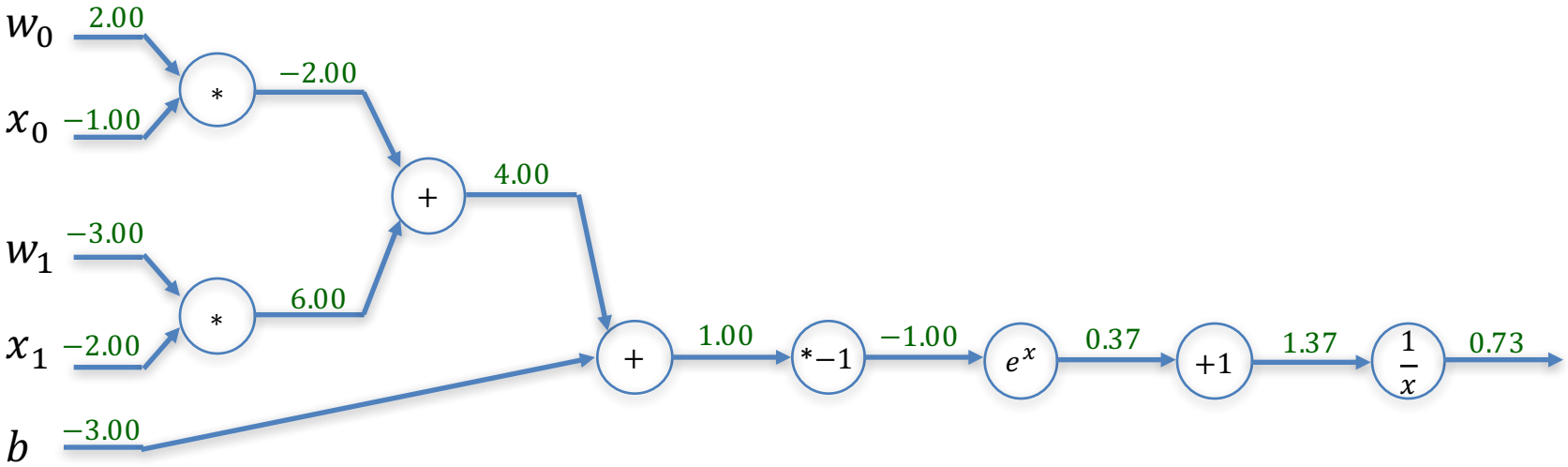
Backprop

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



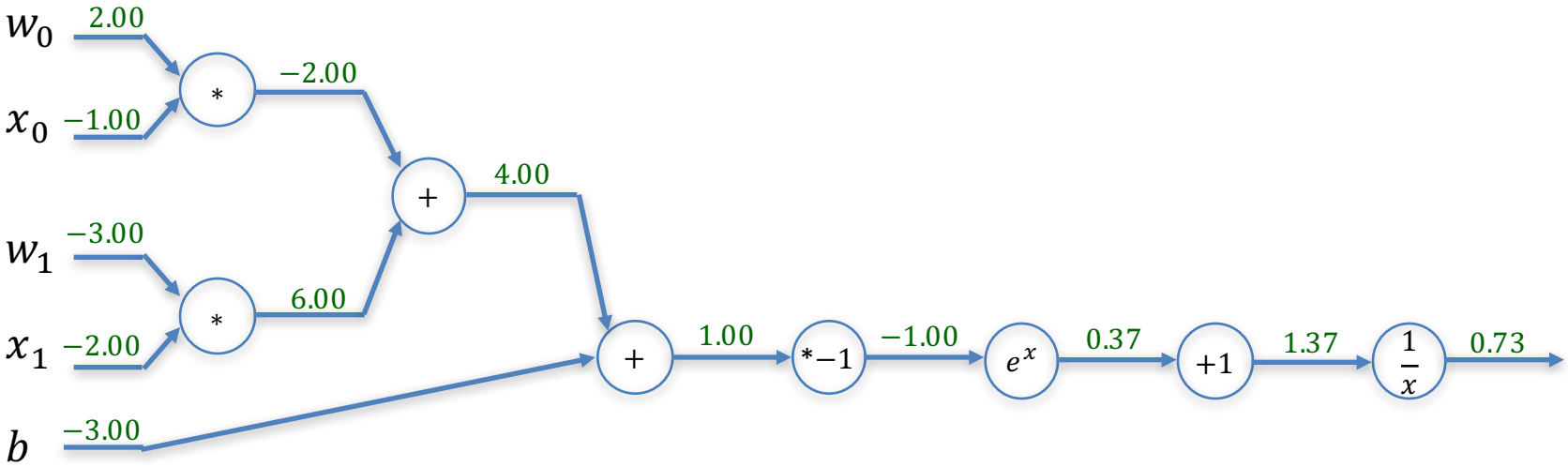
Backprop

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



Backprop

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



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

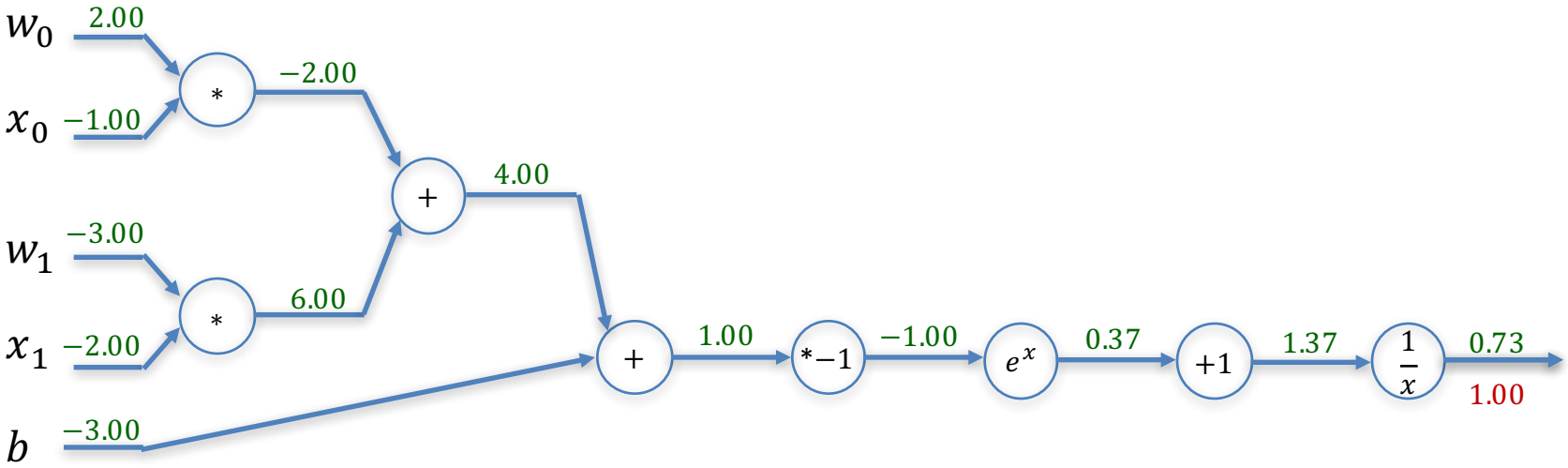
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

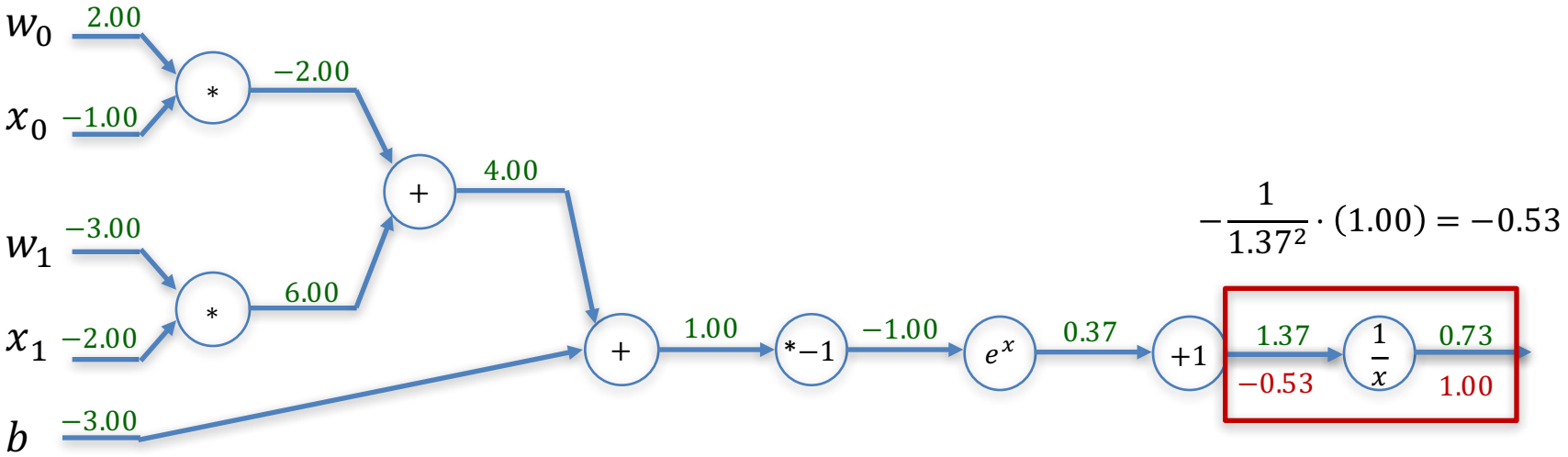
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

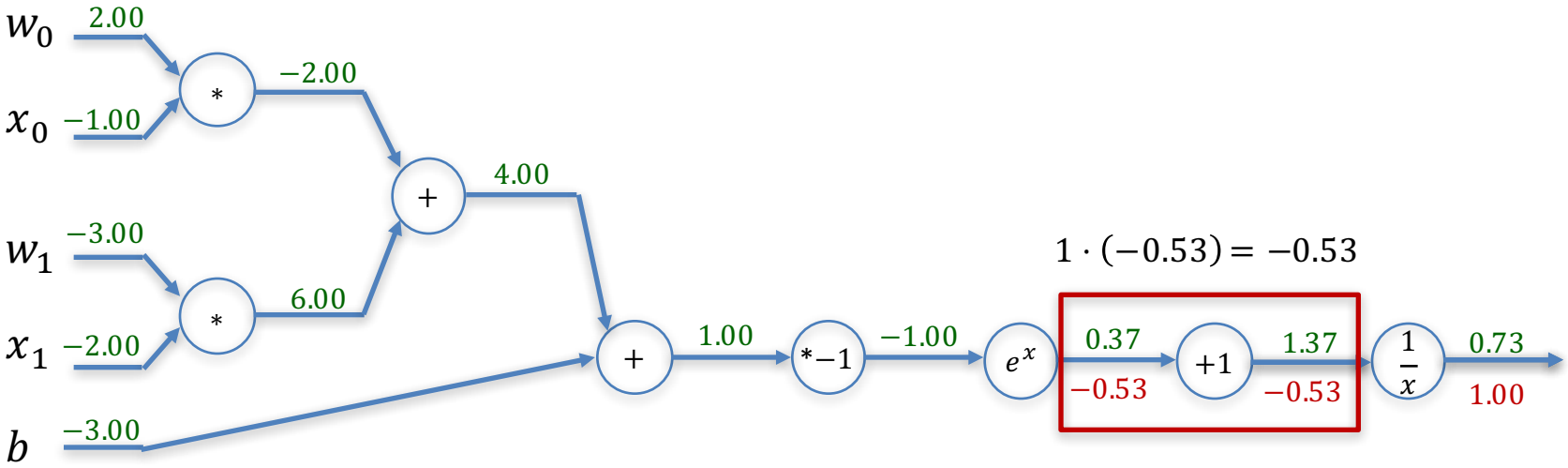
$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

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

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

Backprop

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



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

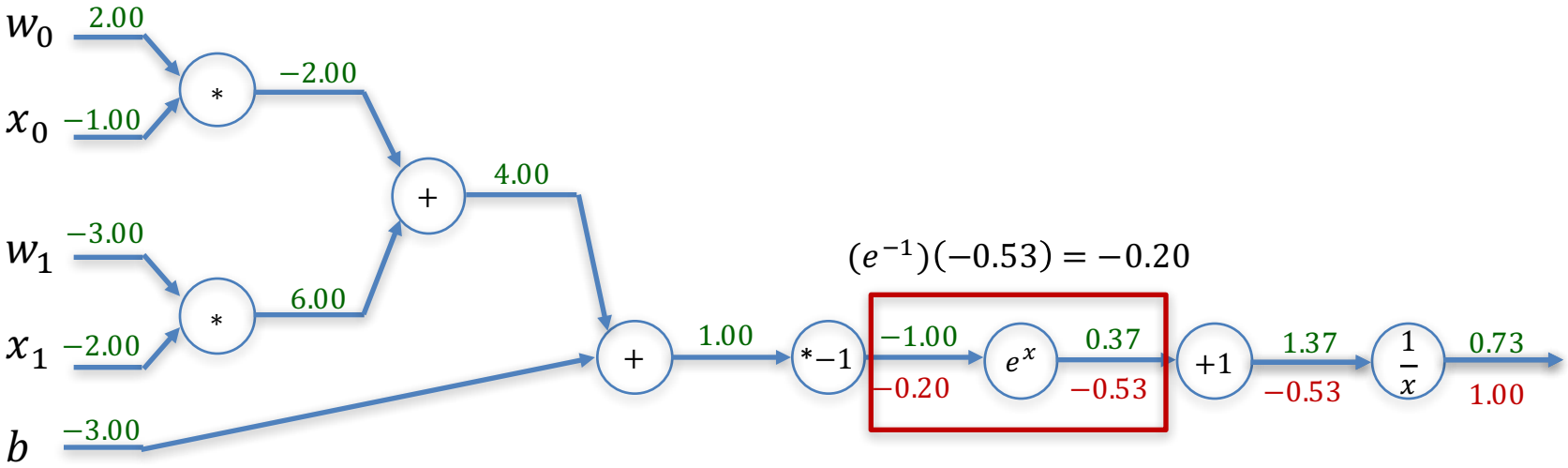
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

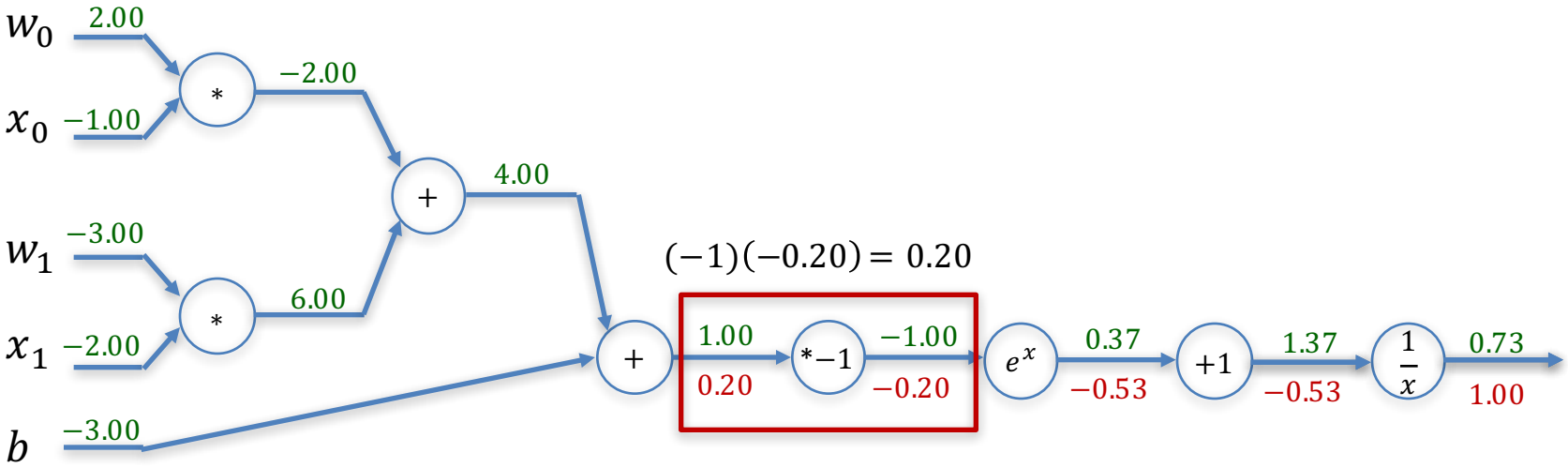
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

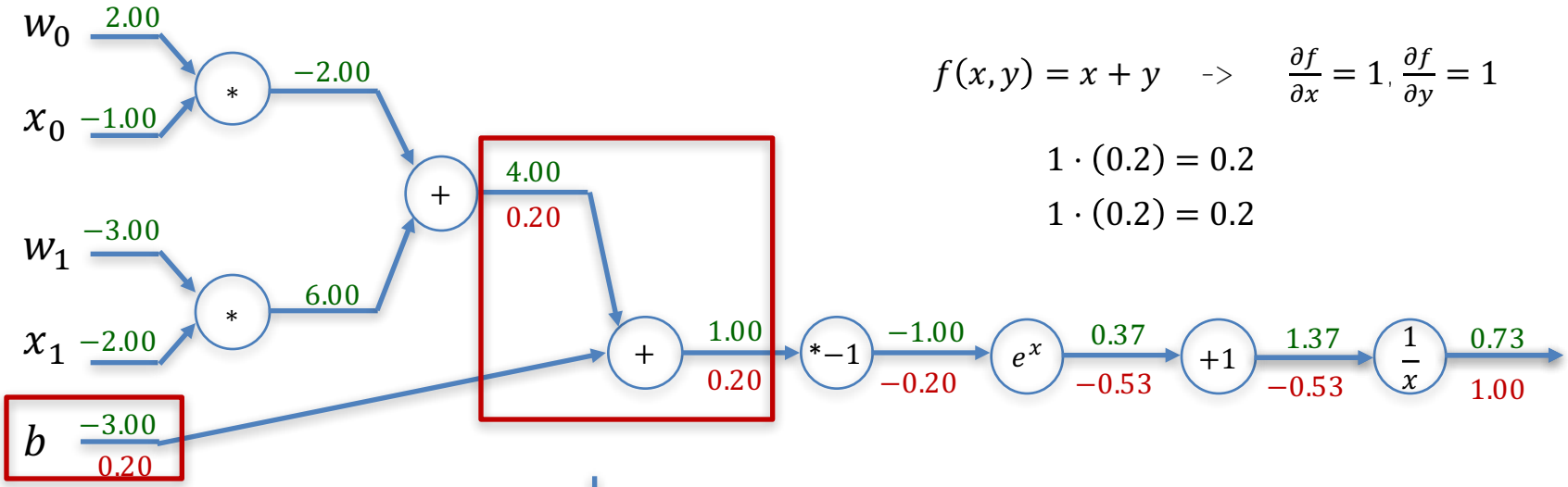
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

$1 \cdot (0.2) = 0.2$

$1 \cdot (0.2) = 0.2$

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

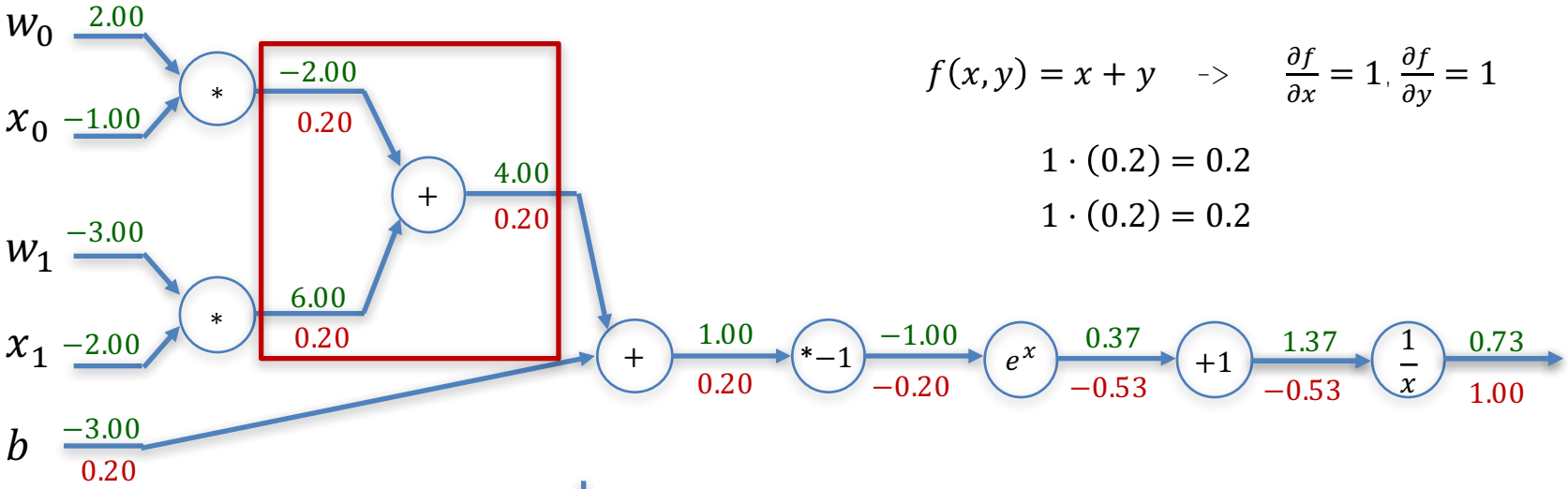
$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

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

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

Backprop

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



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

$1 \cdot (0.2) = 0.2$

$1 \cdot (0.2) = 0.2$

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

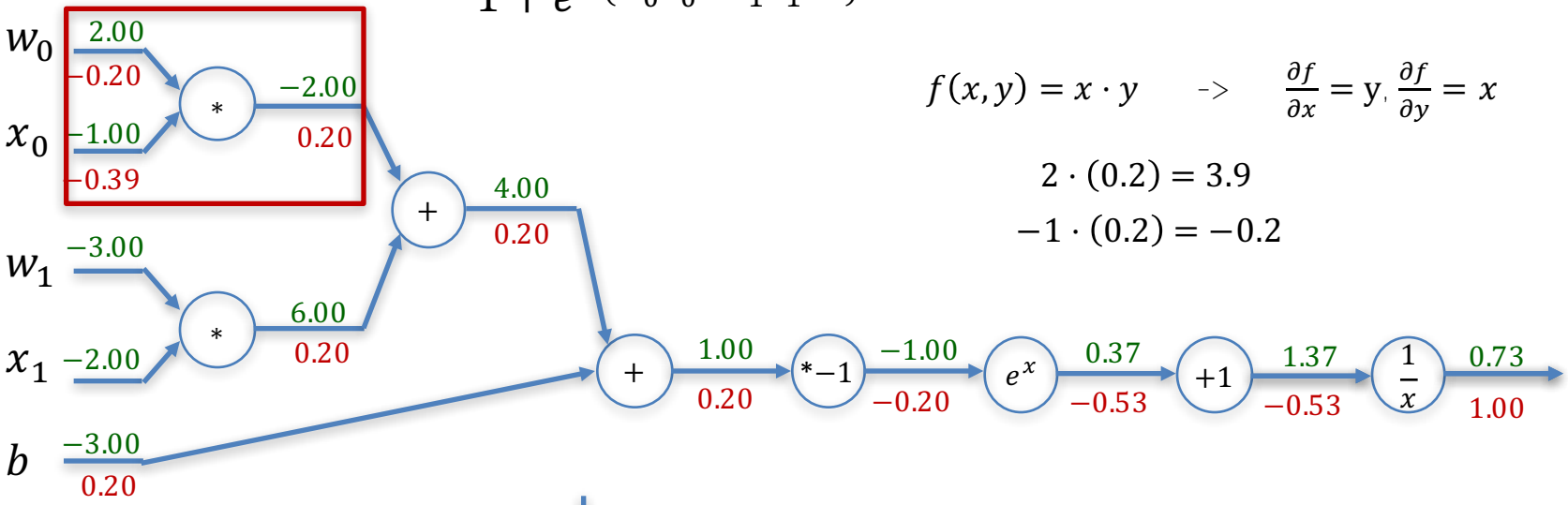
$f_a(x) = ax$ -> $\frac{\partial f_a}{\partial x} = a$

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

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

Backprop

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



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

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

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

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

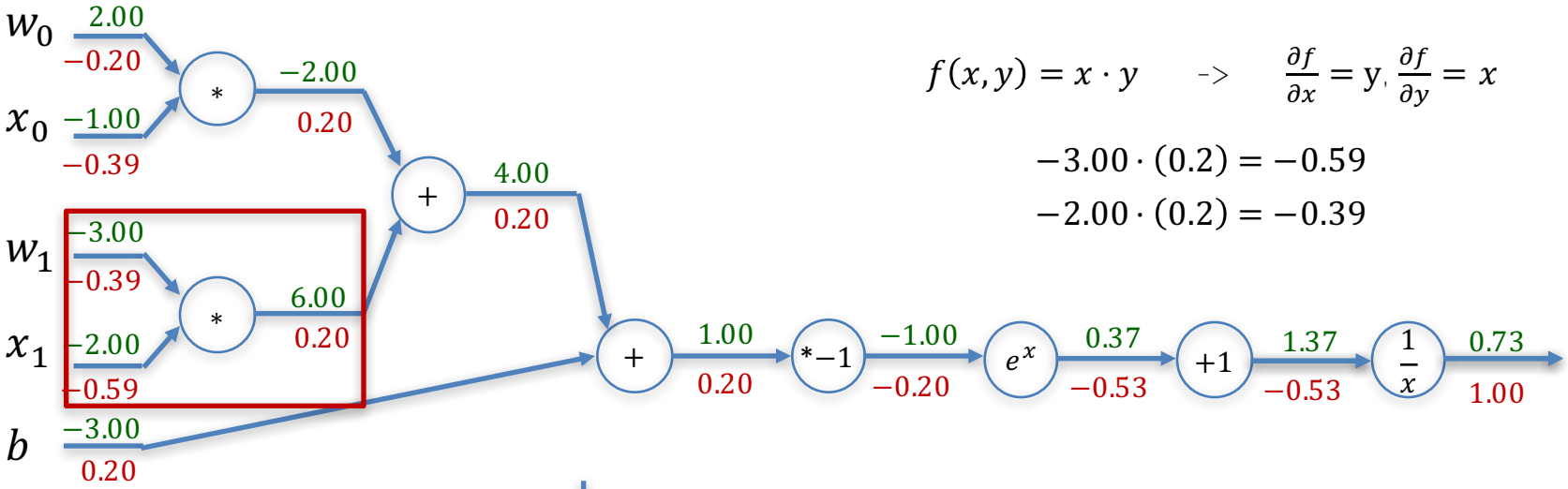
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

Backprop

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



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

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

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

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

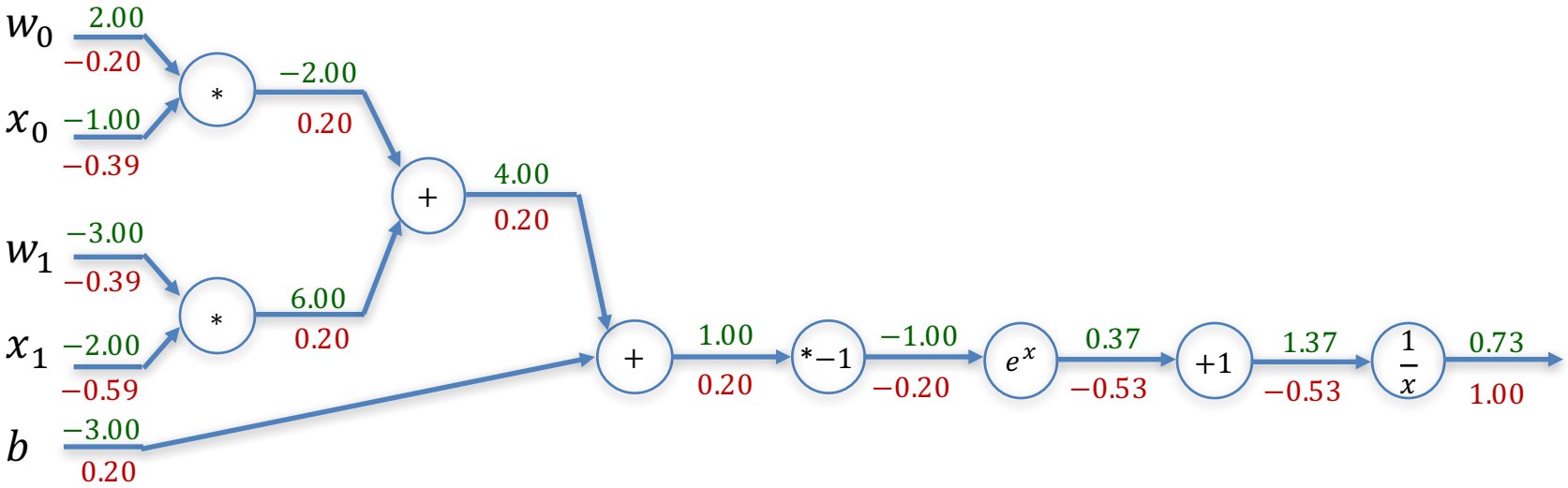
$$f_a(x) = ax \quad \text{->} \quad \frac{\partial f_a}{\partial x} = a$$

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

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

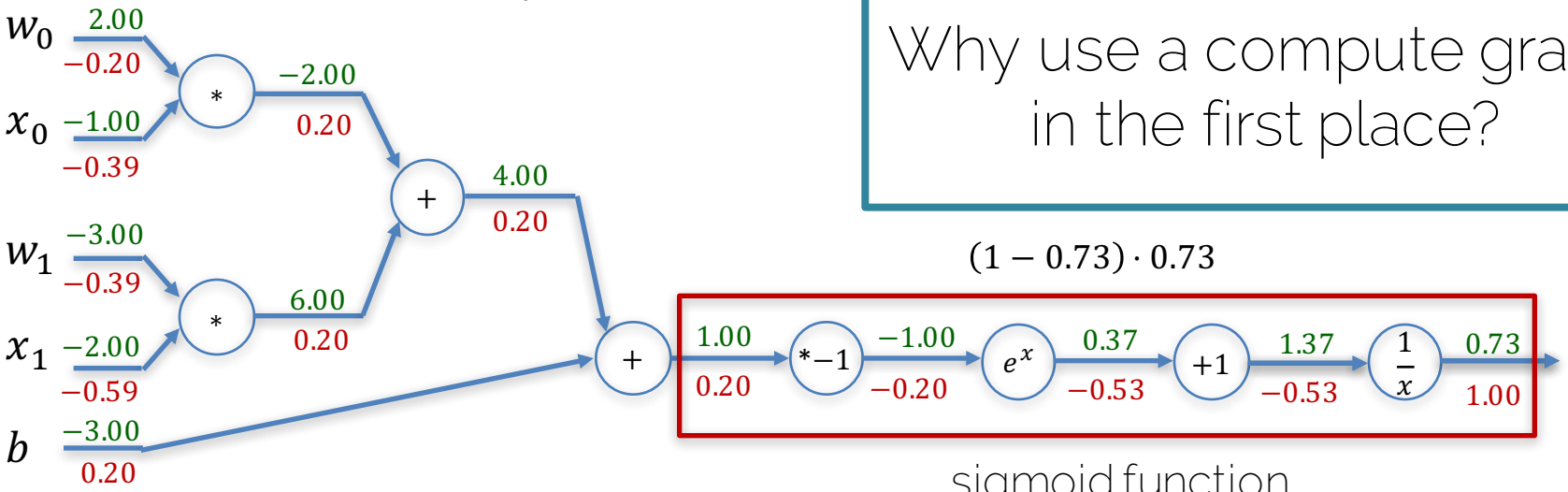
Backprop

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



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?

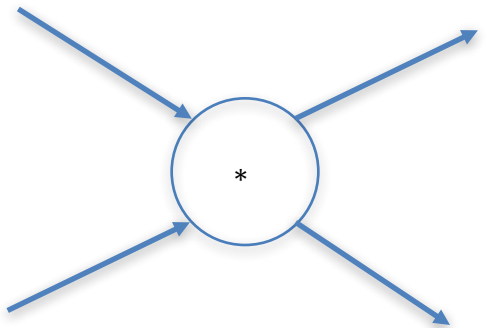
$$(1 - 0.73) \cdot 0.73$$

sigmoid function

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

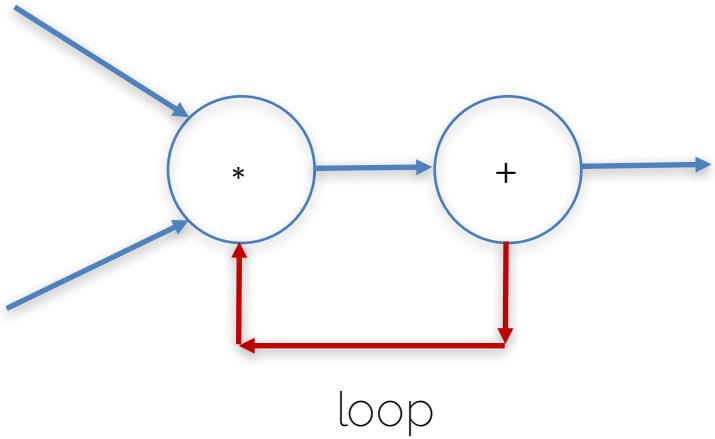
Backpropagation

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

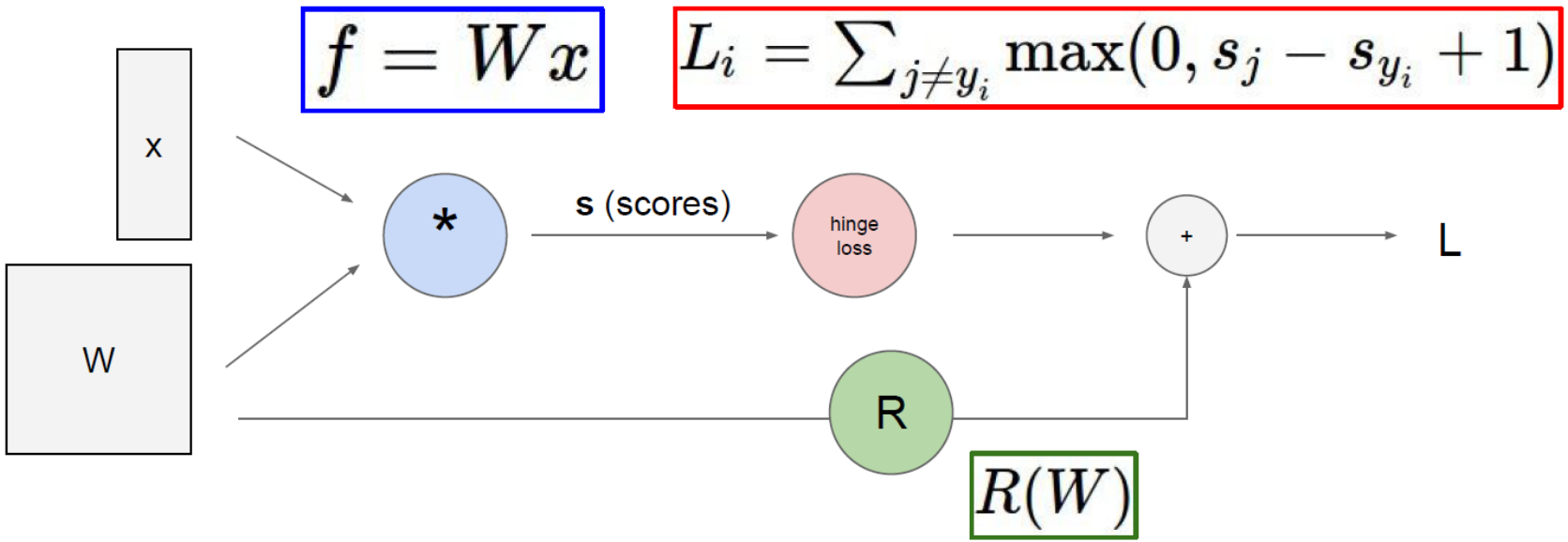


Backpropagation

What happens if there are loops in the graph?



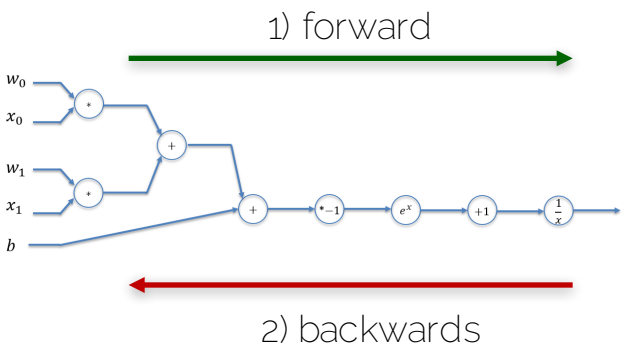
Computational Graph



Combining nodes:
Linear activation node + hinge loss + regularization

Credit Li/Karpathy/Johnson

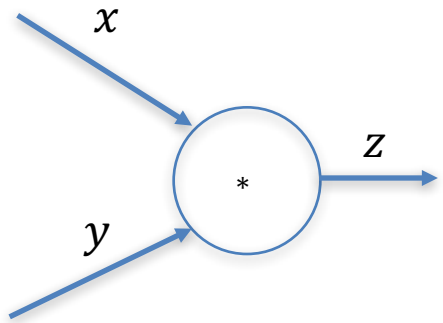
Implementation of Compute Graph




```
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



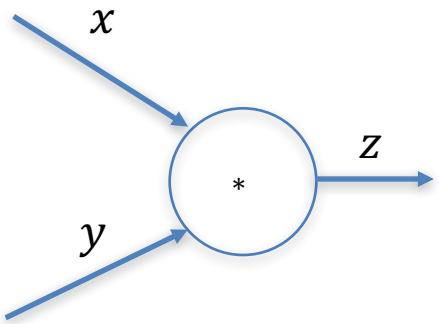

```python
class MulNode(object):
    def forward(x,y):
        z=x*y
        return z
    def backward(dz, x, y):
        dx = y*dz   # [dz/dx * dL/dz]
        dy = x*dz   # [dz/dy * dL/dz]
        return [dx, dy]
```

all values are scalars

Issue?

Implementation of Nodes

- Forward and backward pass of MulNode



all values are scalars

```
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)

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()

Forward()

Backward()

$$f(x) = aX$$

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
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);
    }
  }
}
```

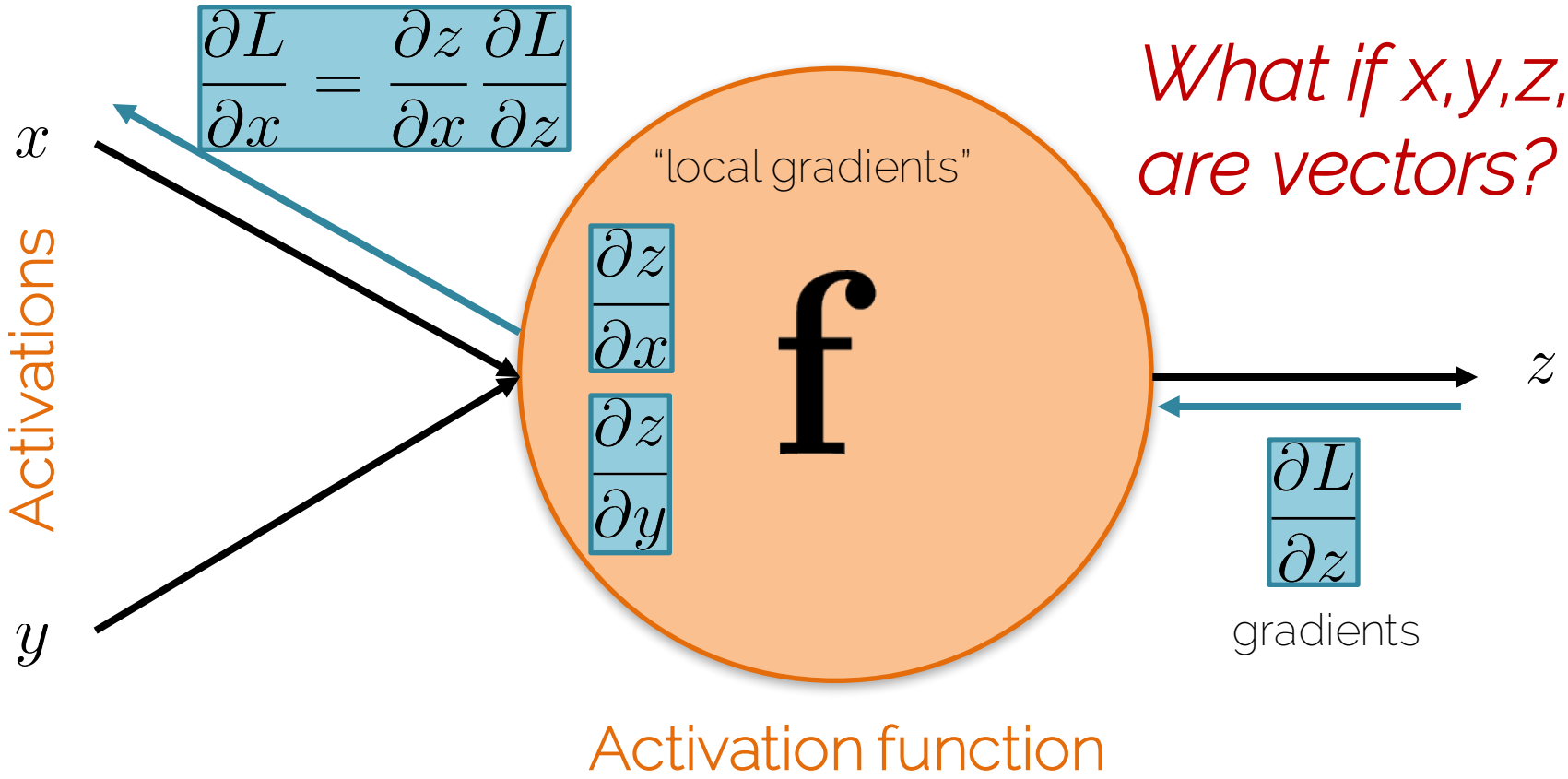
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Forward()

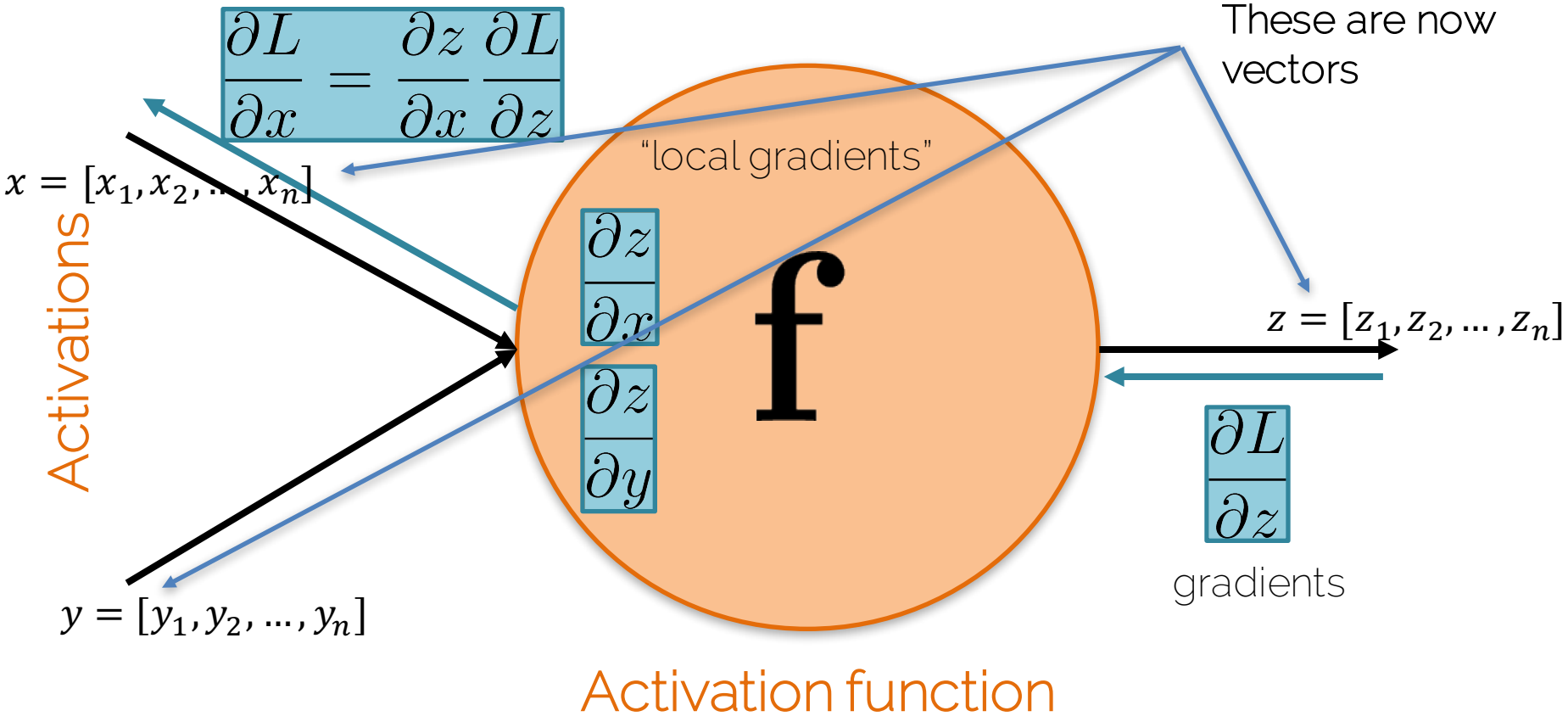
Backward()

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

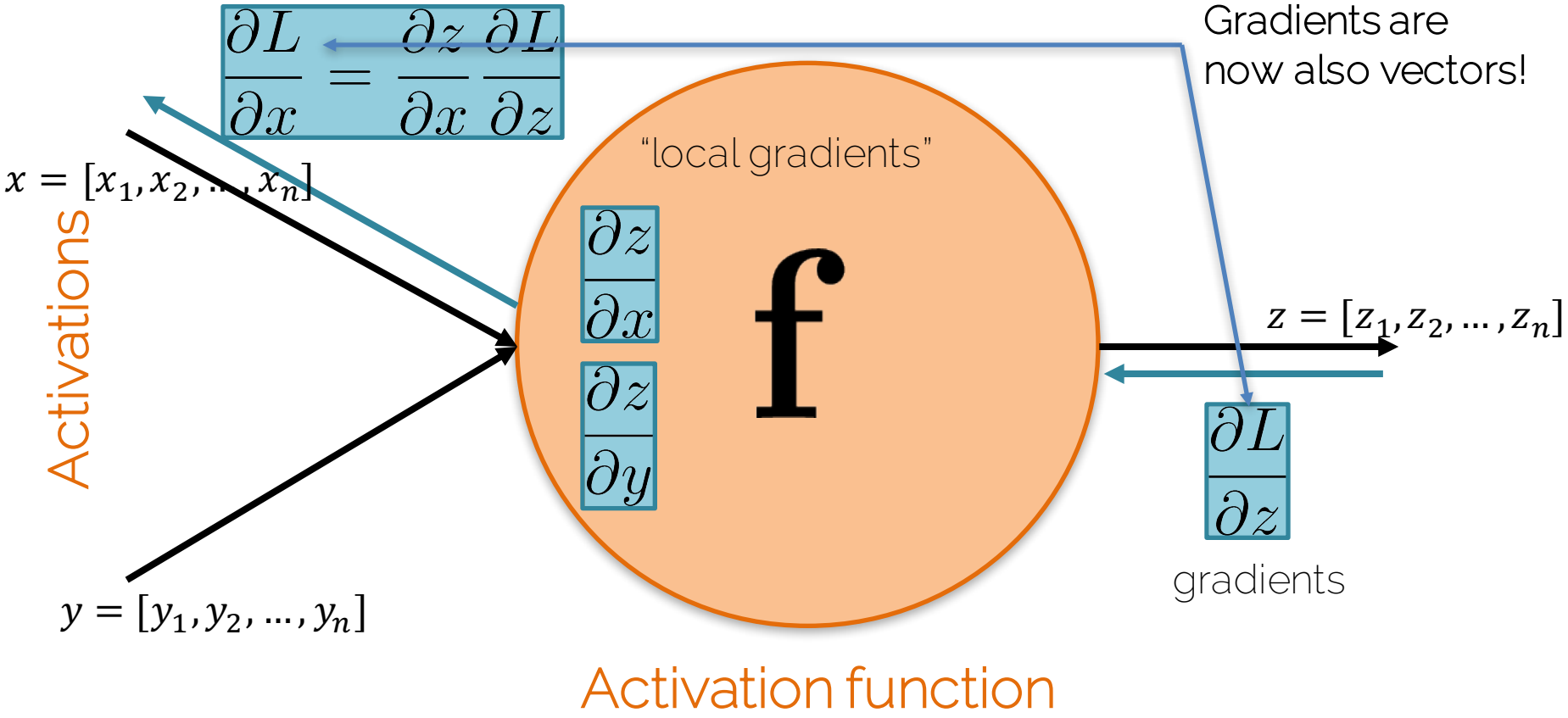
Vectorized Operations



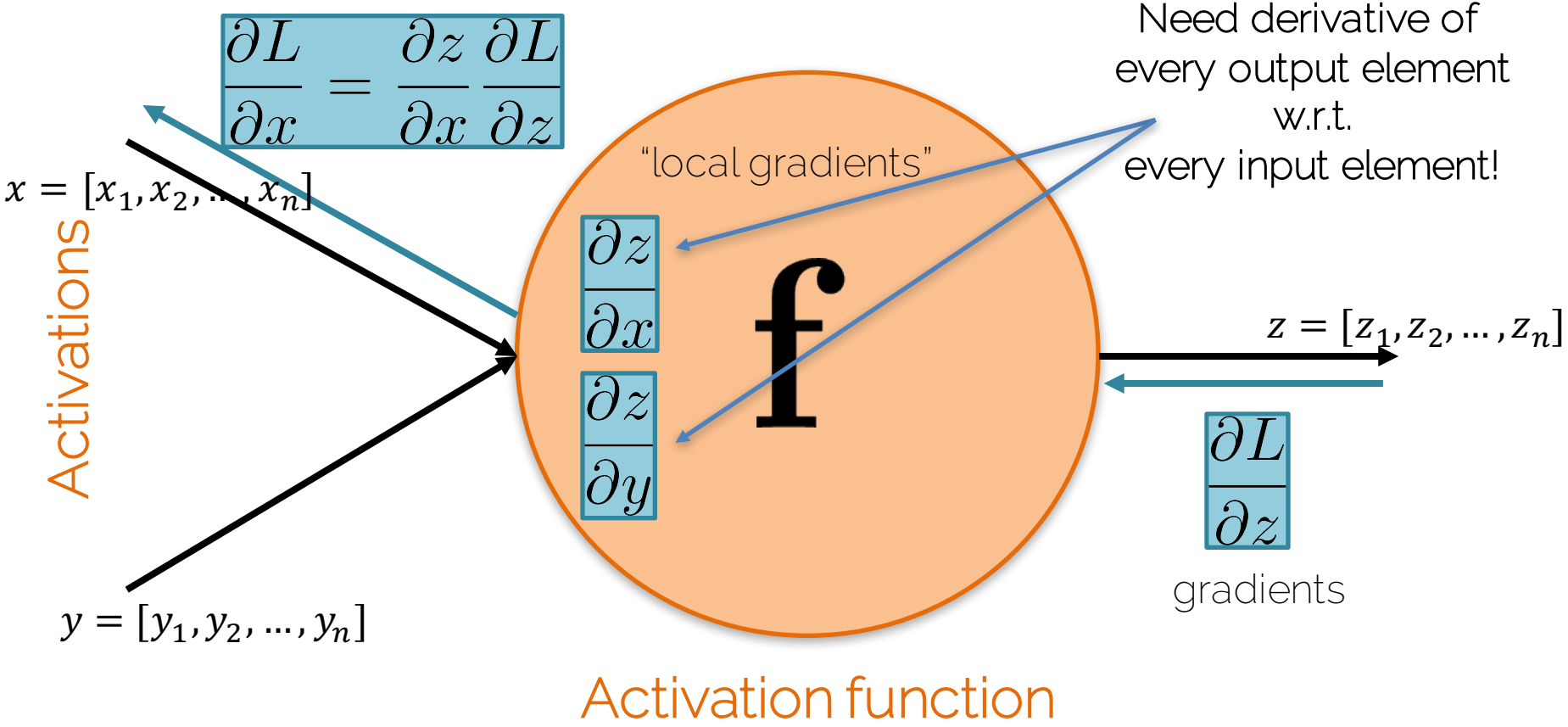
Vectorized Operations



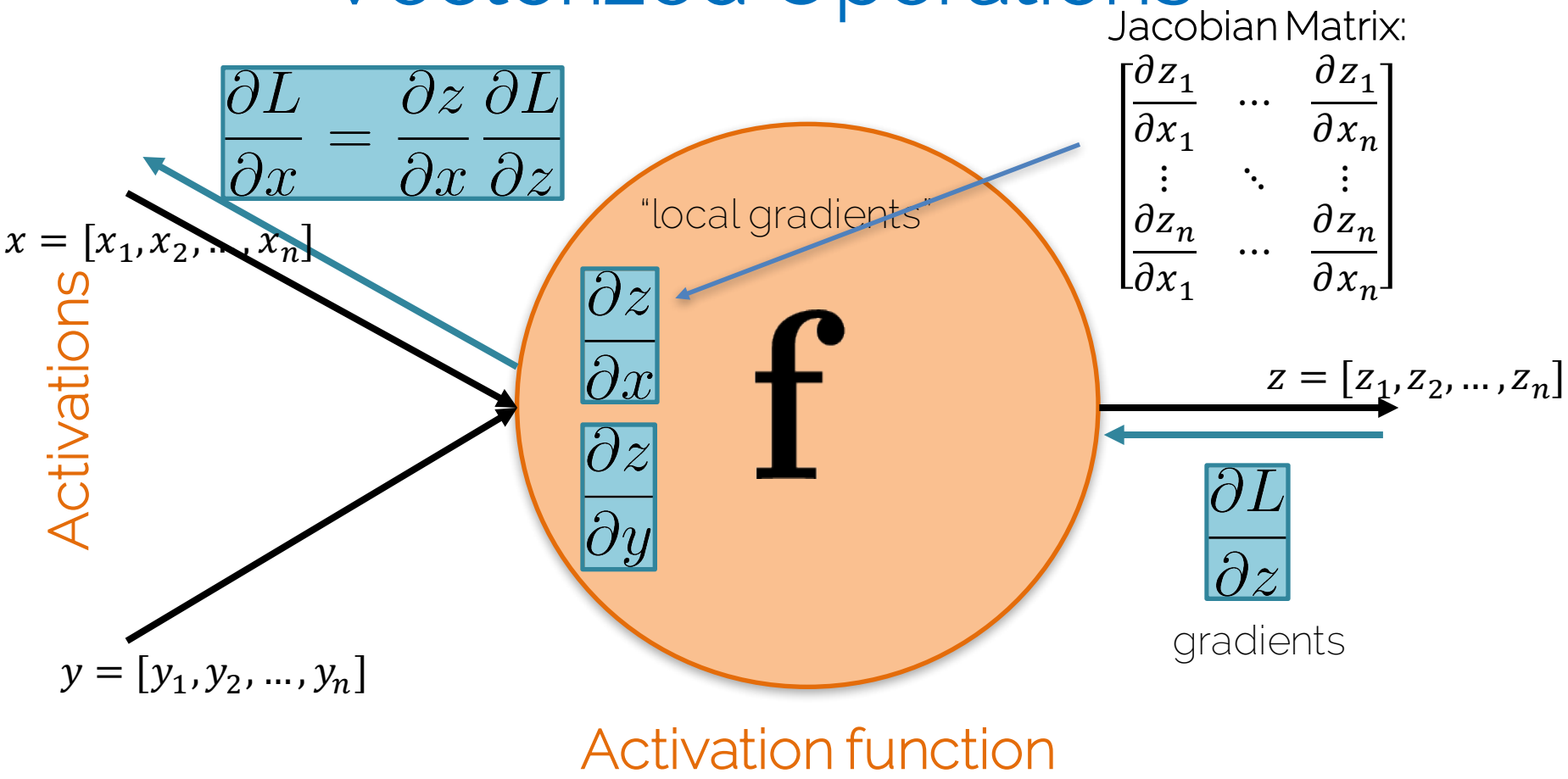
Vectorized Operations



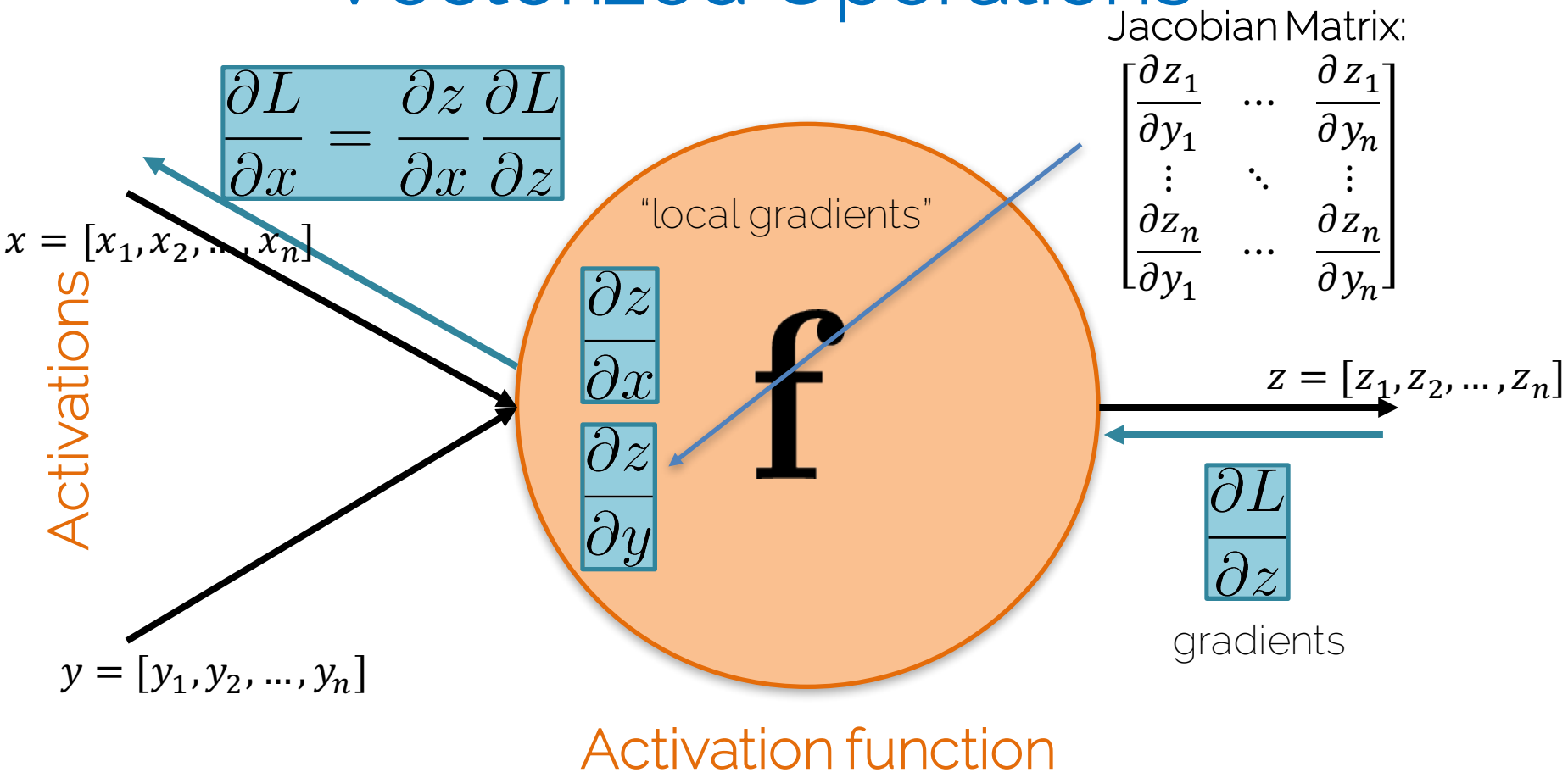
Vectorized Operations



Vectorized Operations



Vectorized Operations

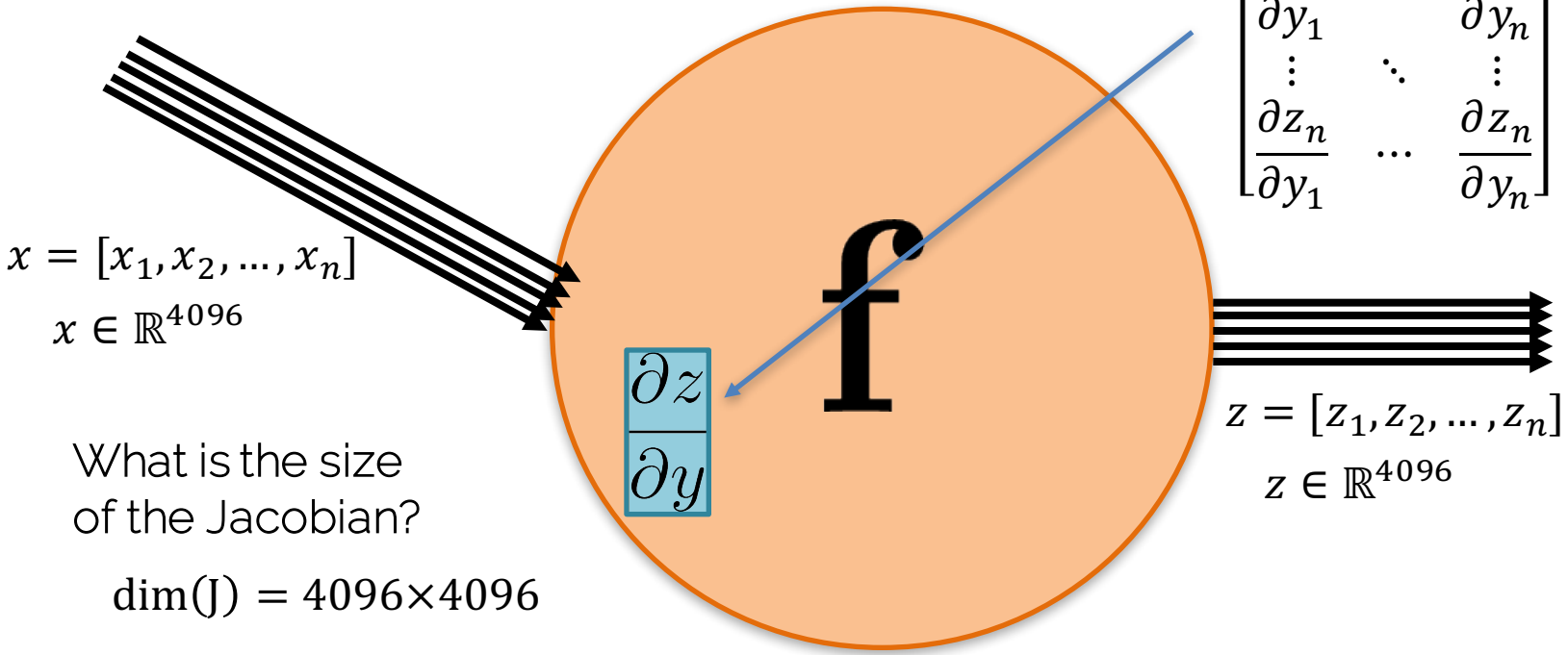


Vectorized Operations

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:

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

Typically, networks are run in batches:

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

How to handle this?

Loss Functions

Naïve Losses

L2 Loss: $L^2 = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$

- Sum of squared differences (SSD)
- Prune to outliers
- Compute-efficient (optimization)
- Optimum is the mean

L1 Loss: $L^1 = \sum_{i=1}^{n} |y_i - f(x_i)|$

- Sum of absolute differences
- Robust
- Costly to compute
- Optimum is the median

12	24	42	23
34	32	5	2
12	31	12	31
31	64	5	13

x_i

15	20	40	25
34	32	5	2
12	31	12	31
31	64	5	13

y_i

$$L^2(x, y) = 9 + 16 + 4 + 4 + 0 + \cdots + 0 = 66$$

$$L^1(x, y) = 3 + 4 + 2 + 2 + 0 + \cdots + 0 = 15$$

Hinge Loss (SVM Loss)

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

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
	Loss	2.9		

$L_1 =$
$\max(0, 5.1 - 3.2 + 1) +$
$\max(0, -1.7 - 3.2 + 1) =$
$= \max(0, 2.9) + \max(0, -3.9) =$
$= 2.9 + 0 =$
$= \mathbf{2.9}$

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
	Loss	2.9	0	

$$L_2 =$$
$$\max(0, 1.3 - 4.9 + 1) +$$
$$\max(0, 2.0 - 4.9 + 1) =$$
$$= \max(0, -2.6) + \max(0, -1.9) =$$
$$= 0 + 0 =$$
$$= \mathbf{0}$$

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
Loss		2.9	0	10.9

$L_3 =$
$\max(0, 2.2 - (-3.1) + 1) +$
$\max(0, 2.5 - (-3.1) + 1) =$
$= \max(0, 5.3) + \max(0, 5.6) =$
$= 5.3 + 5.6 =$
$= \mathbf{10.9}$

Hinge Loss (SVM Loss)

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

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
	Loss	2.9	0	10.9

Full Loss (over all pairs):

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i =$$

$$= \frac{L_1 + L_2 + L_3}{3} =$$

$$= \frac{2.9 + 0 + 10.9}{3} =$$

$$= \mathbf{4.6}$$

Weight Regularization & SVM Loss

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$

Full loss $L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max\left(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1\right) + \lambda R(W)$

L^1-reg: $R^1(W) = \sum_{i=1}^{N} \sum_{j \neq y_i} |w_i|$

L^2-reg: $R^2(W) = \sum_{i=1}^{N} \sum_{j \neq y_i} w_i^2$

Weight Regularization & SVM Loss

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$

Full loss $L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max\left(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1\right) + \lambda R(W)$

L^1-reg: $R^1(W) = \sum_{i=1}^{N} \sum_{j \neq y_i} |w_i|$

L^2-reg: $R^2(W) = \sum_{i=1}^{N} \sum_{j \neq y_i} w_i^2$

Example:

$x = [1,1,1,1]$

$w_1 = [1, 0, 0, 0]$

$w_2 = [0.25, 0.25, 0.25, 0.25]$

$w_1^T x = w_2^T x = 1$

$R^2(w_1) = 1$

$R^2(w_2) = 0.25^2 + 0.25^2 + 0.25^2 + 0.25^2 = 0.25$

Cross-Entropy (Softmax)

Softmax $$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}}\right)$$

Score function $$s = f(x_i, W)$$

e.g., $$f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

Cross-Entropy (Softmax)

Softmax $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

3.2
5.1
-1.7

Loss

Cross-Entropy (Softmax)

Softmax $$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}}\right)$$

Score function $$s = f(x_i, W)$$

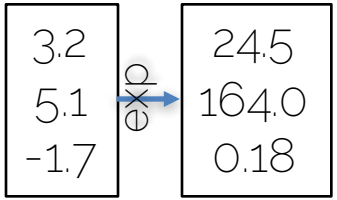
e.g., $$f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1



Loss

Cross-Entropy (Softmax)

Softmax $$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}}\right)$$

Score function $$s = f(x_i, W)$$

e.g., $$f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

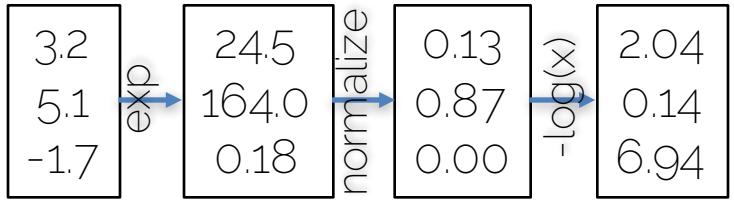
Cross-Entropy (Softmax)

Softmax $$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}}\right)$$

Score function $$s = f(x_i, W)$$

e.g., $$f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$$

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)



Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
	Loss	2.04	0.14	6.94

Cross-Entropy (Softmax)

Softmax $$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}}\right)$$

Score function $$s = f(x_i, W)$$

e.g., $$f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$$

Suppose: 3 training examples and 3 classes



scores				
	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
Loss		2.04	0.14	6.94

Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i =$$

$$= \frac{L_1 + L_2 + L_3}{3} =$$

$$= \frac{2.04 + 0.14 + 6.94}{3} =$$

$$= \mathbf{9.12}$$

Hinge Loss vs Softmax

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

Softmax: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Hinge Loss vs Softmax

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

Softmax: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:

$s = [5, -3, 2]$

$s = [5, 10, 10]$

$s = [5, -20, -20]$

$y_i = 0$

Hinge Loss vs Softmax

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

Softmax: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:

	Hinge loss:
$s = [5, -3, 2]$	$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$
$s = [5, 10, 10]$	$\max(0, 10 - 5 + 1) + \max(0, 10 - 5 + 1) = 12$
$s = [5, -20, -20]$	$\max(0, -20 - 5 + 1) + \max(0, -20 - 5 + 1) = 0$

$y_i = 0$

Hinge Loss vs Softmax

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

Softmax: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:	Hinge loss:	Softmax loss:
$s = [5, -3, 2]$	$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$	Google… -ln(e^(5)/(e^(5)+e^(-3)+e^(2)))=0.05
$s = [5, 10, 10]$	$\max(0, 10 - 5 + 1) + \max(0, 10 - 5 + 1) = 12$	Google… -ln(e^(5)/(e^(5)+e^(10)+e^(10)))=5.70
$s = [5, -20, -20]$	$\max(0, -20 - 5 + 1) + \max(0, -20 - 5 + 1) = 0$	Google… -ln(e^(5)/(e^(5)+e^(-20)+e^(-20)))=2.e-11

$y_i = 0$

Hinge Loss vs Softmax

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

Softmax: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:	Hinge loss:	Softmax loss:
$s = [5, -3, 2]$	$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$	Google… -ln(e^(5)/(e^(5)+e^(-3)+e^(2)))=0.05
$s = [5, 10, 10]$	$\max(0, 10 - 5 + 1) + \max(0, 10 - 5 + 1) = 12$	Google… -ln(e^(5)/(e^(5)+e^(10)+e^(10)))=5.70
$s = [5, -20, -20]$	$\max(0, -20 - 5 + 1) + \max(0, -20 - 5 + 1) = 0$	Google… -ln(e^(5)/(e^(5)+e^(-20)+e^(-20)))=2.e-11

$y_i = 0$

Softmax *always* wants to improve!
Hinge Loss saturates

Loss in Compute Graph

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

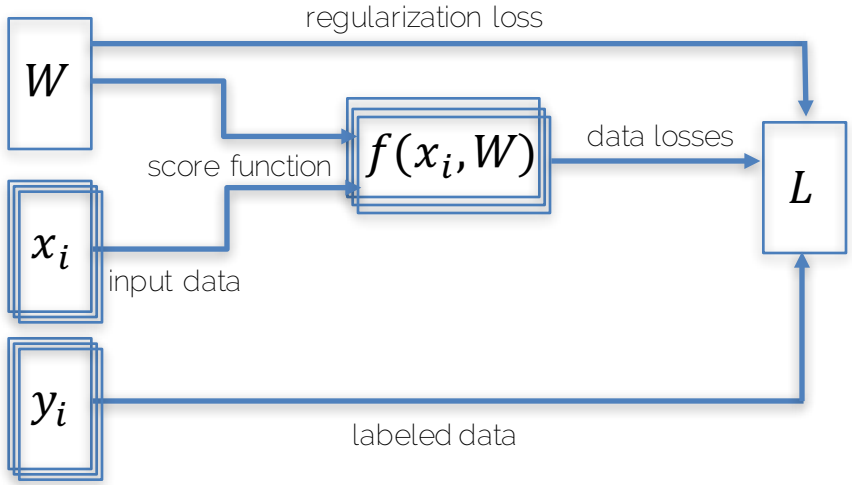
Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

Softmax $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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

e.g., L^2-reg: $R^2(W) = \sum_{i=1}^{N} w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^{N} L_i + R^2(W)$



Compute Graphs

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Softmax $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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

e.g., L^2-reg: $R^2(W) = \sum_{i=1}^{N} w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^{N} L_i + R^2(W)$

Compute Graphs

Score function $s = f(x_i, W)$

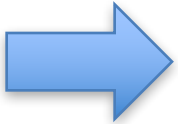
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

Softmax $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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

e.g., L^2-reg: $R^2(W) = \sum_{i=1}^{N} w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^{N} L_i + R^2(W)$



Want to find optimal W. I.e., weights are unknowns of optimization problem

Compute gradient w.r.t. W.

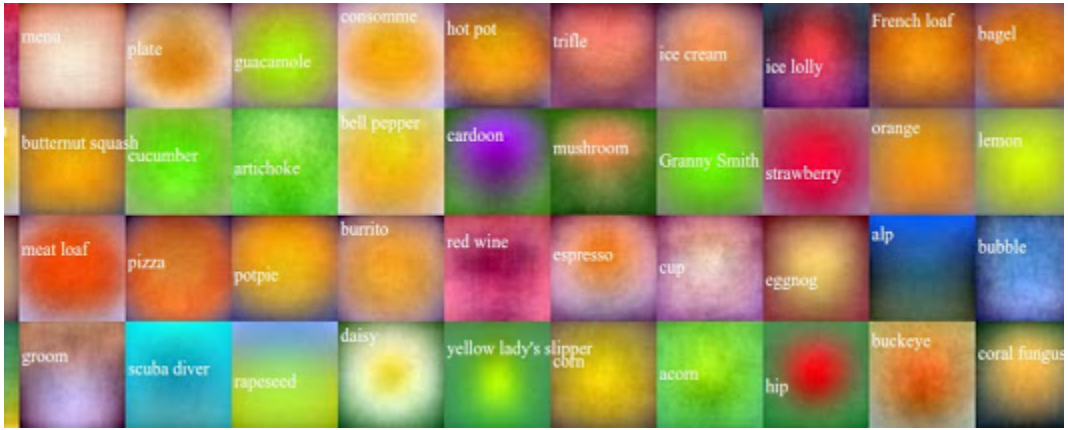
Gradient $\nabla_W L$ is computed via backpropagation

Introduction to Neural Networks

Neural Network

- Linear score function $f = Wx$

On CIFAR-10



On ImageNet

Credit Li/Karpathy/Johnson

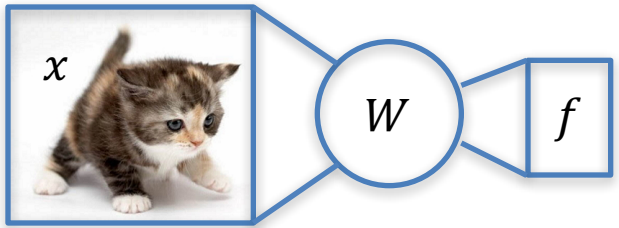
Neural Network

- Linear score function $f = Wx$

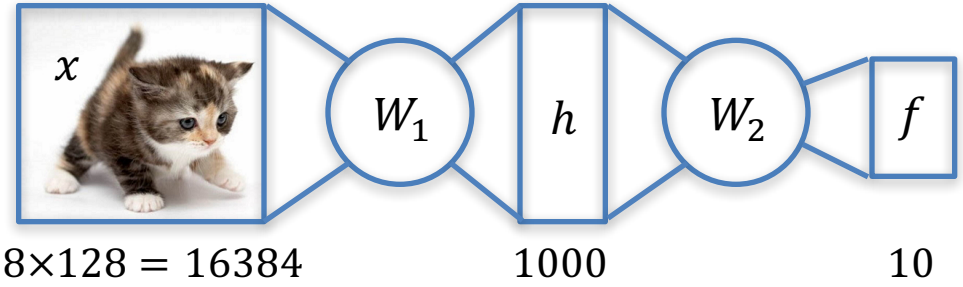
- Neural network is a nesting of 'functions'
  - 2-layers: $f = W_2 \max(0, W_1 x)$
  - 3-layers: $f = W_3 \max(0, W_2 \max(0, W_1 x))$
  - 4-layers: $f = W_4 \tanh(\mathrm{W}_3, \max(0, W_2 \max(0, W_1 x)))$
  - 5-layers: $f = W_5 \sigma(W_4 \tanh(\mathrm{W}_3, \max(0, W_2 \max(0, W_1 x))))$
  - … up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

Neural Network

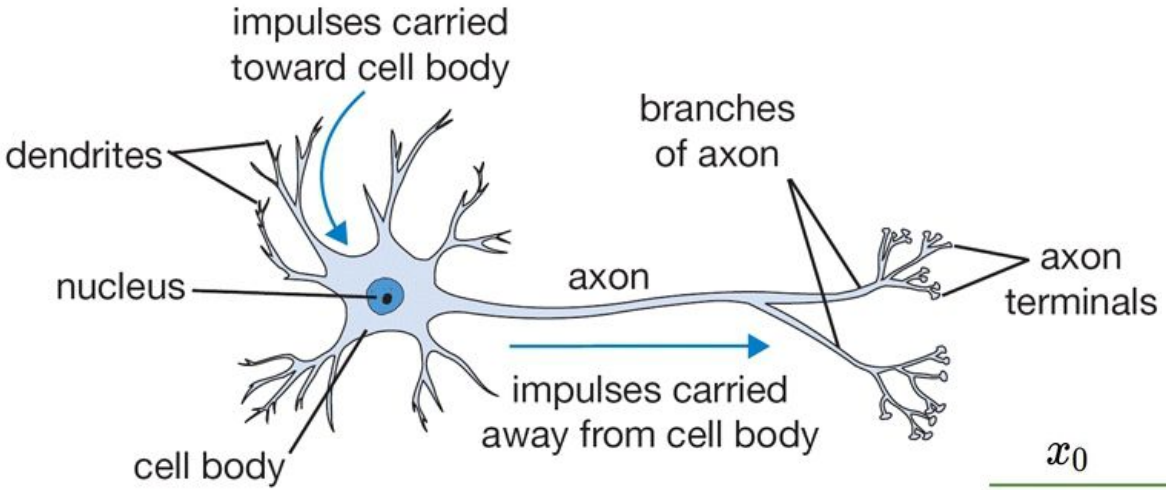
1-layer network: $f = Wx$

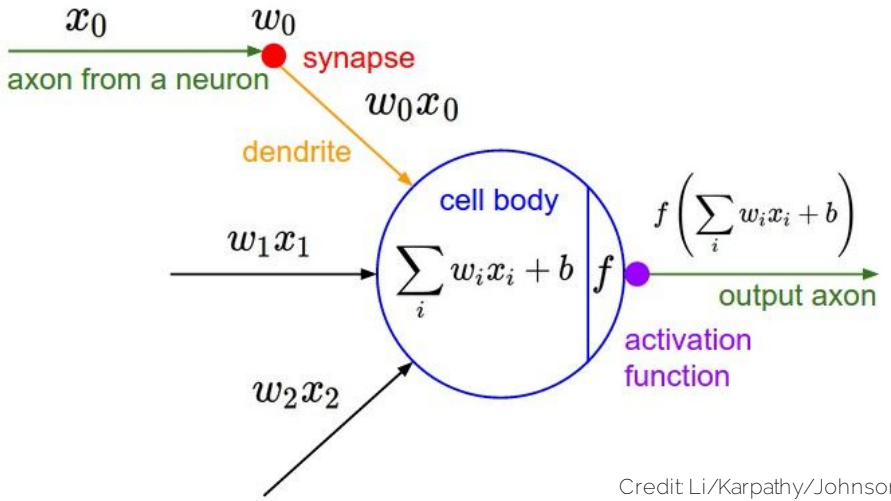


2-layer network: $f = W_2 \max(0, W_1 x)$



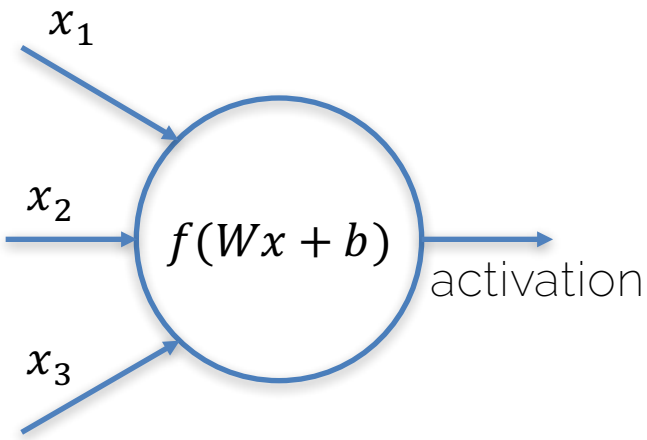
Neurons





Credit Li/Karpathy/Johnson

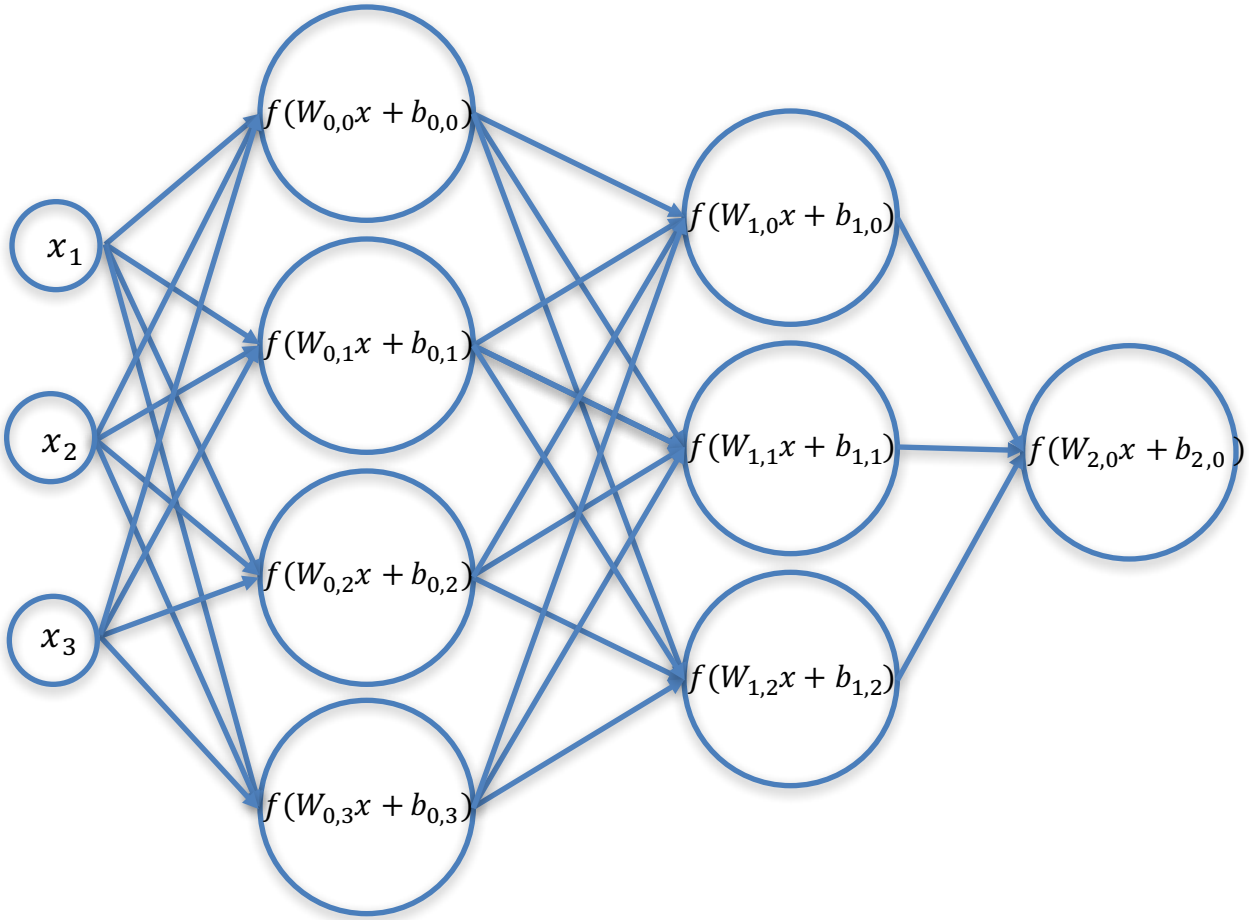
Neurons



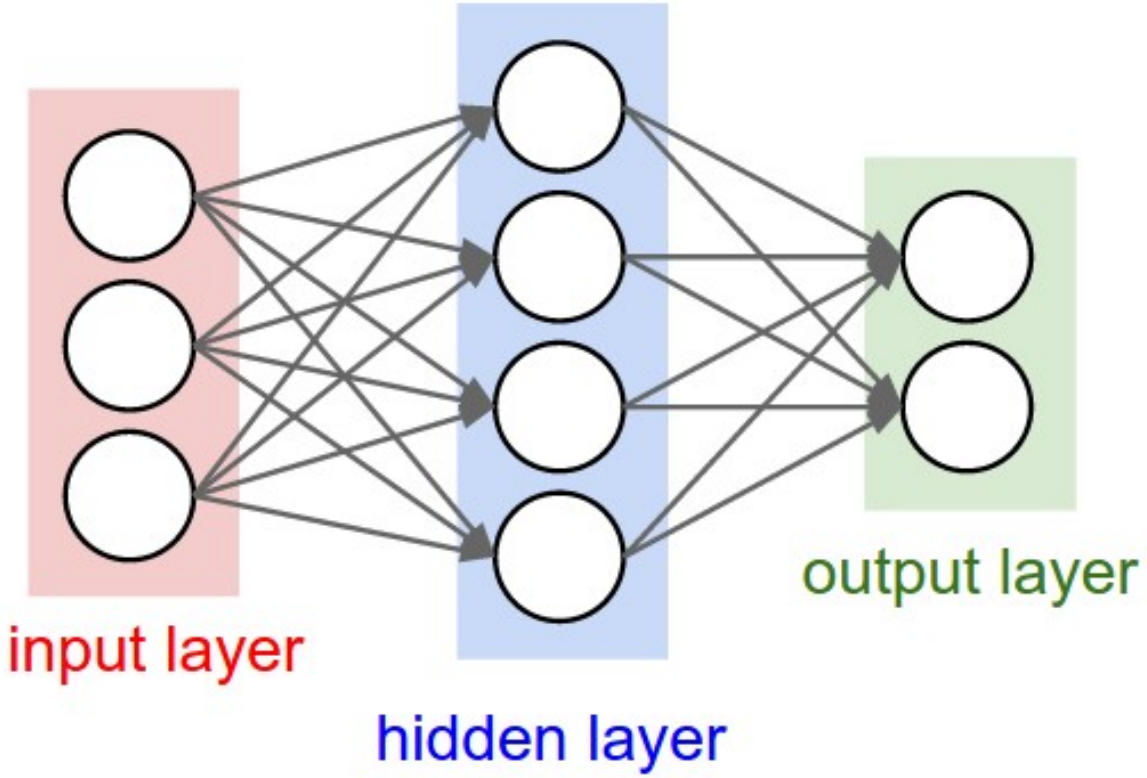
Linear function: $Wx + b$
Non-linearity (activation: $f(x)$

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

Net of Neurons

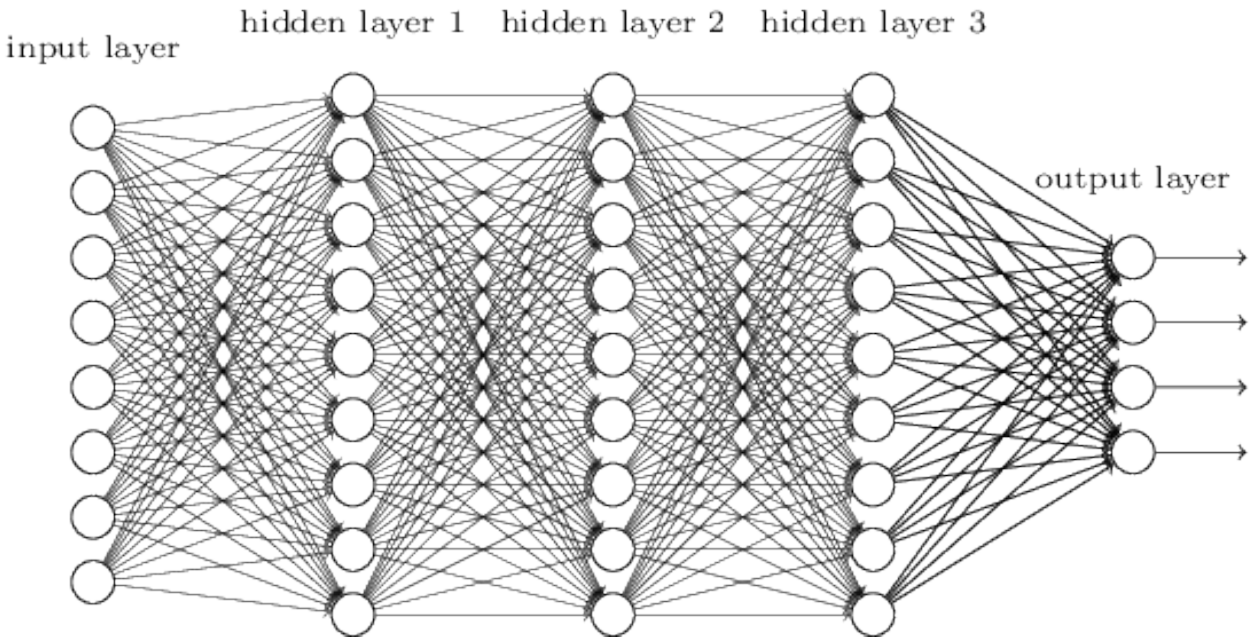


Neural Network



Credit Li/Karpathy/Johnson

Neural Network



Neural Network

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

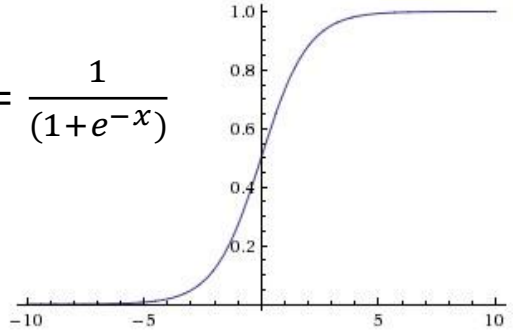
Why activation functions?

Why not just concatenate?

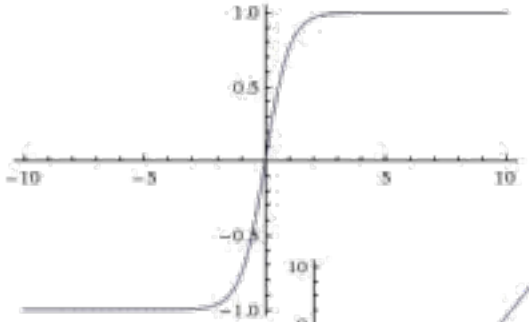
Would be much cheaper to compute….

Activation Functions

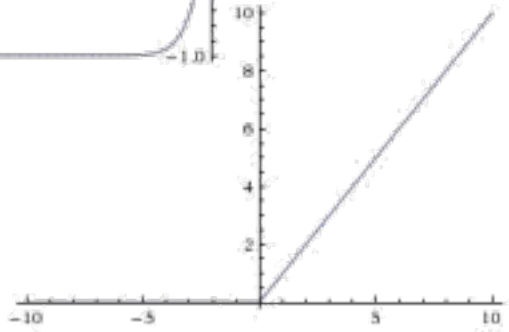
Sigmoid: $\sigma(x) = \frac{1}{(1+e^{-x})}$



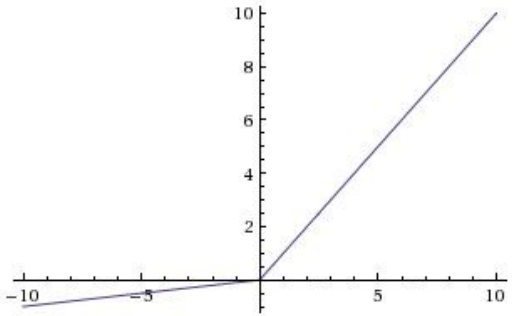
tanh: $\tanh(x)$



ReLU: $\max(0, x)$



Leaky ReLU: $\max(0.1x, x)$



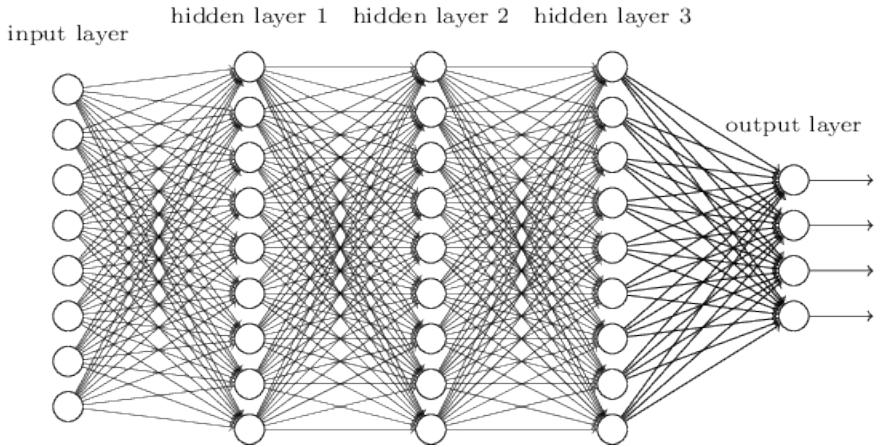
Parametric ReLU: $\max(\alpha x, x)$

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

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 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 ☺

Administrative Things

- Next Thursday 18th of May:
  - More on Neural Networks ☺
  - More theory, best practices, applications
- Tomorrow: 1st Q&A session of the first assignment
- Tentative date for the exam: 18th of August
  (not confirmed though)