19COC102

Introduction to Neural Networks

Dr. Andrea Soltoggio

Overview of the lecture

This lecture will focus on

- Discussion on last week's exercises
- Presentation of the coursework
- Some biology and inspiration for neural networks
- Basic principles of artificial neural networks (ANN)
- Perceptron learning
- Supervised learning: backpropagation of error

Coursework

- Presentation of coursework
- Due 16th March at 12:00AM, submission via Learn
- To submit: report, Jupyter notebook
- Use latex template
- Accurately cite sources

- General comments:
  - First year for this new coursework
  - Generally simple so most students will find it easy
  - Flexible in terms of implementation task and style
  - However, score a high mark it is necessary to have:
    - a clear presentation of techniques, a choice of neural architectures and parameters that illustrates differences in the results, precision in the description, accuracy, clarity, and good formatting (e.g. understandable figures)

Coursework - Useful links

- Managing environments in Conda: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

- Example of Jupyter Notebook: https://github.com/rickiepark/pytorch-examples/blob/master/mnist.ipynb

- Pytorch tutorial https://github.com/yunjey/pytorch-tutorial

- Latex template for the report https://www.overleaf.com/read/gkphfttddwfc

- Note: for insights on deep networks architectures, next week's lecture will focus on CNNs.

Inspiration - the biological neuron

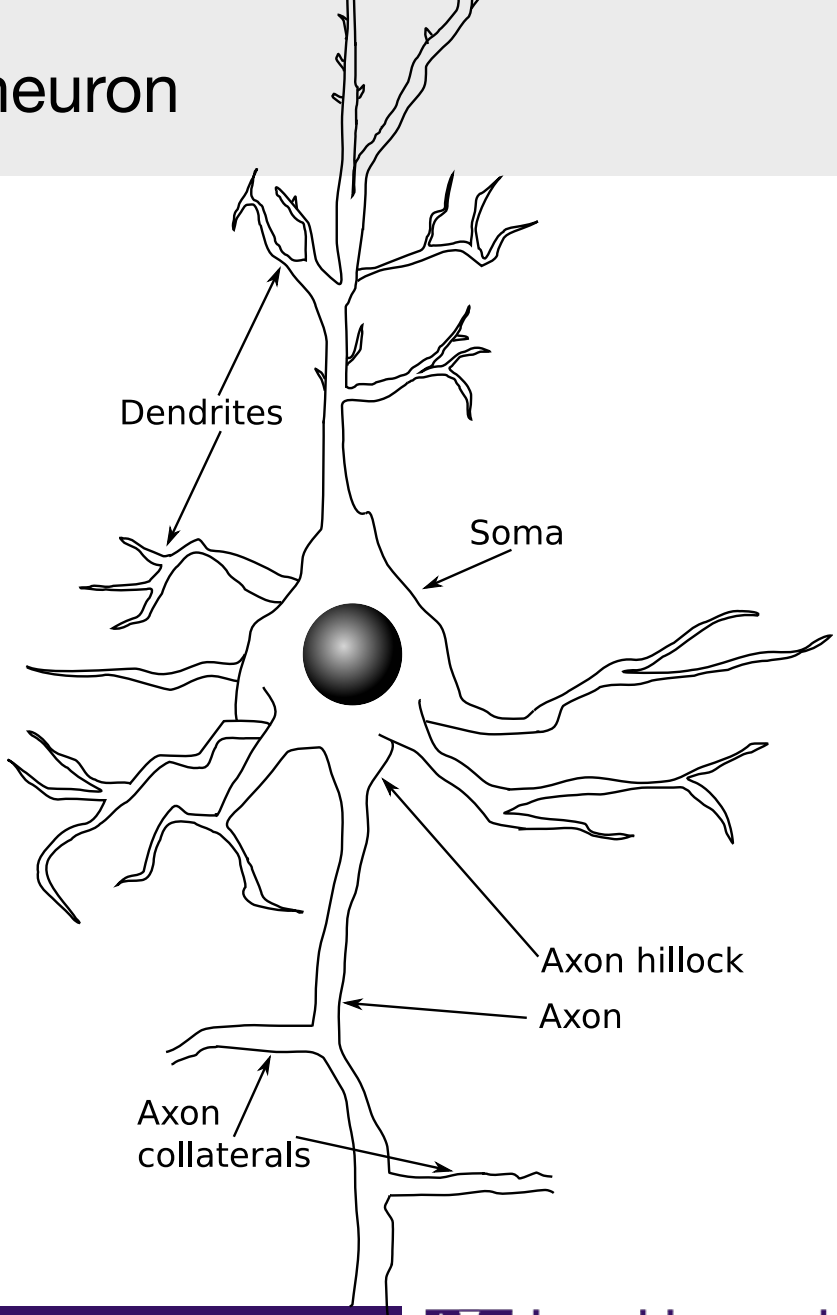
Simplified drawing of a neuron cell.

The image was drawn after the examples in (Bear et al., 2005).

In the human brain, there are approximately 100 billion neurons.

The some in the neuron triggers a chain reaction along the axon called **action potential** that propagates along the axon and causes the release of neuro-transmitters at the synaptic terminals.

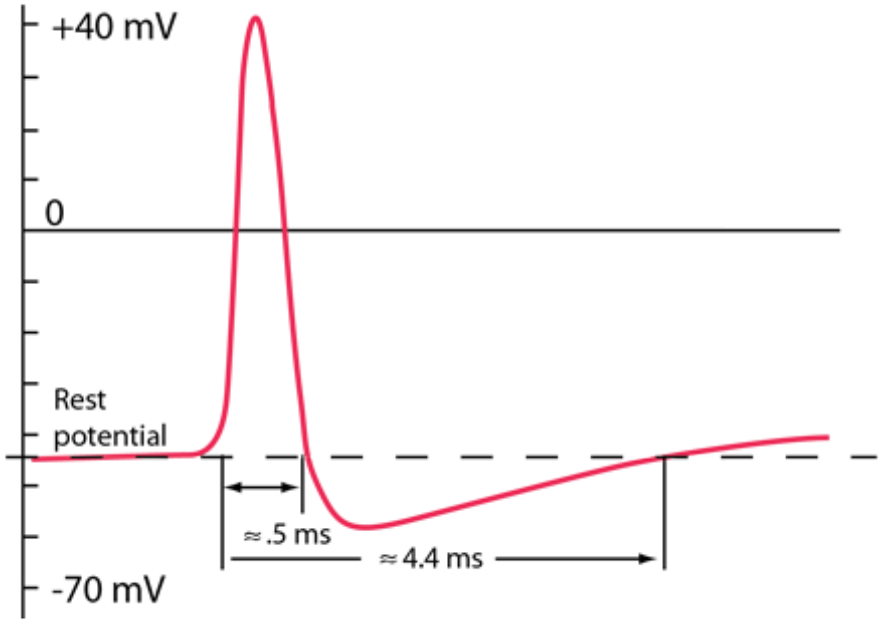
Each neuron on average has 1000 synapses, thus, there are approximately 100 trillion synapses in the human brain.



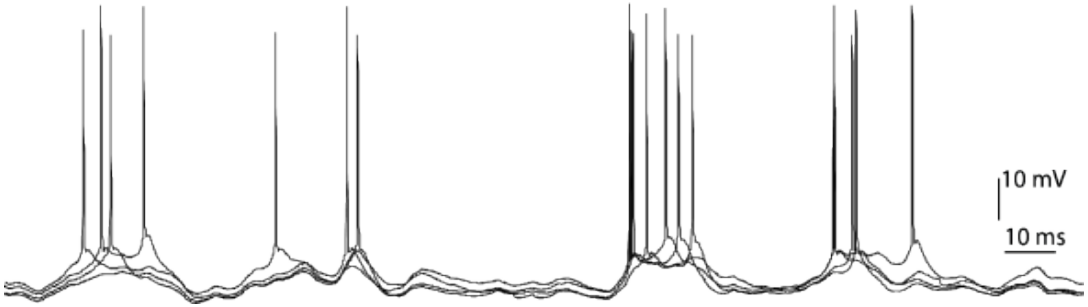
Inspiration - action potential (spikes)

Action potential. The voltage on the neuron's membrane changes as an action potential travels along the axon.

After a neuron has fired an action potential, it cannot fire another one immediately but has to wait to 'recharge': this is called refractory period.



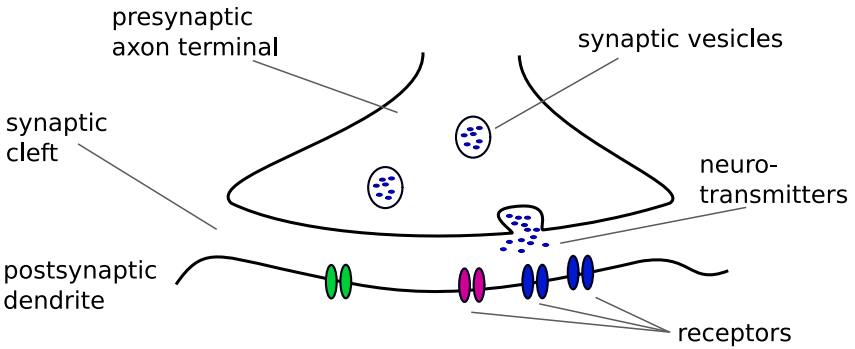
Neurons can fire with particular frequencies, or in bursts, according to a variety of patterns.



Inspiration - Synapses and types of neurons

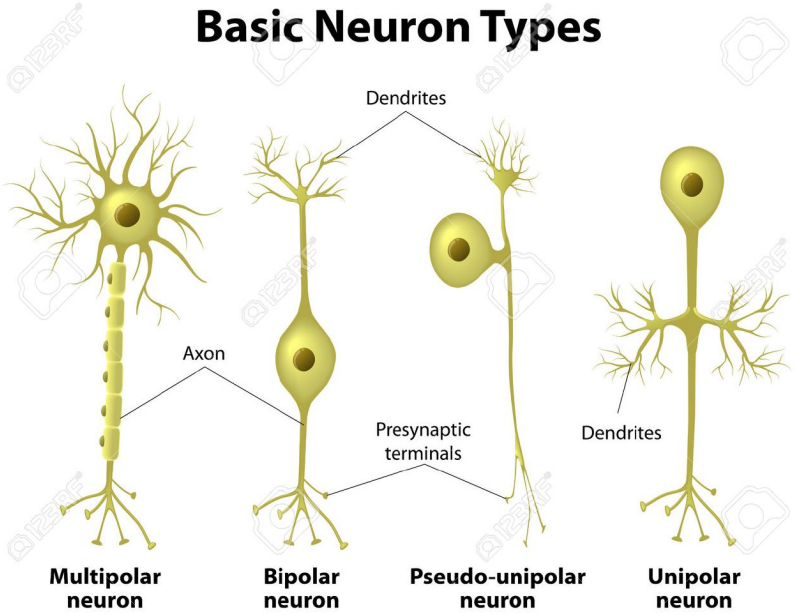
Once a neuro-trasmitter is released due an action potential, it travels through the synaptic cleft and reaches the receptors.

Neuro-transmitters are then destroyed or re-absorbed so that the effect is limited the brief time of action potential.



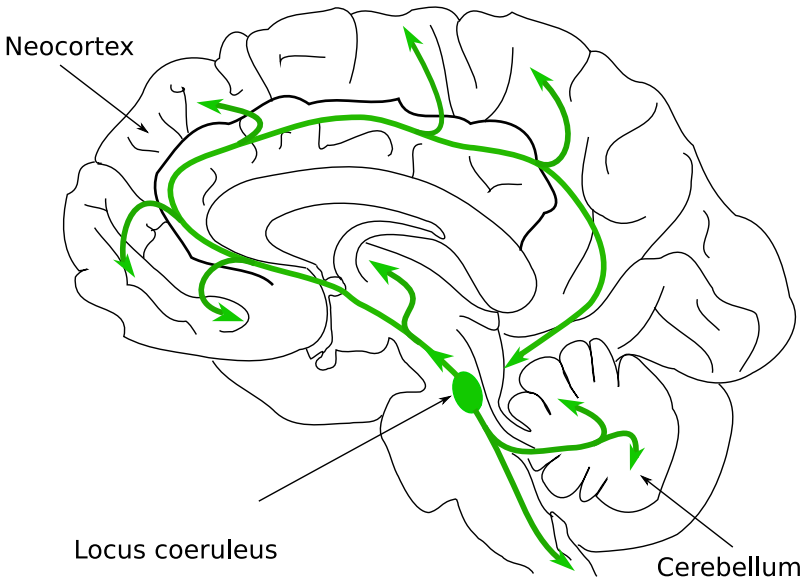
There are many different types of neurons that transmit many different types of neurotransmitters.

E.g. glutamate (Glu), gamma-aminobutyric acid (GABA), N-metyl-D-aspartate (NMDA). Some of these are excitatory signals, some are inhibitory signals, i.e. they excite or inhibit the post-synaptic neurons.

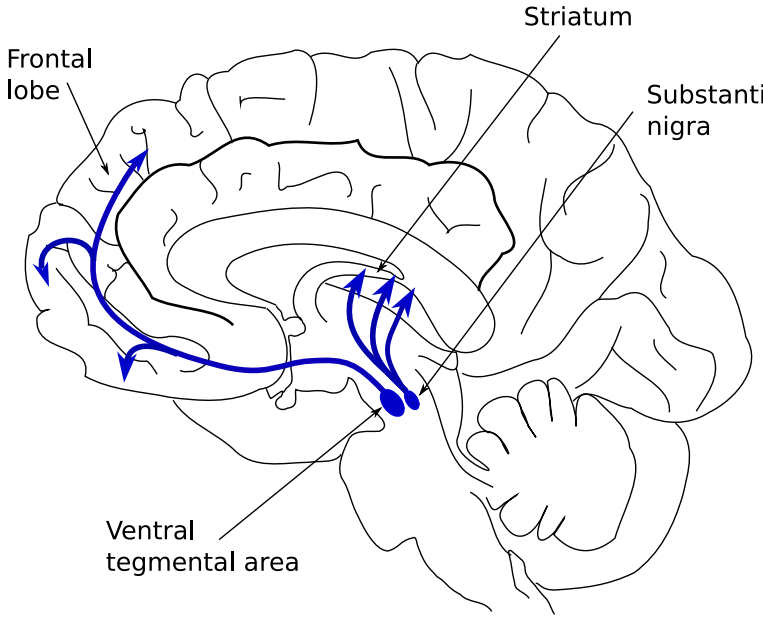


Inspiration - Other neurotransmitters

There are other chemicals produced by special neurons that play a role in attention, memory consolidation, and reward learning. There are called modulatory neurons and release chemicals such as **dopamine**, **acetylcholine**, **norephineprine**, **serotonin**.



noradrenergic system



dopaminergic system

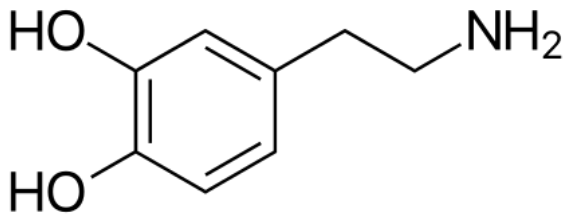
Interesting facts - Dopamine

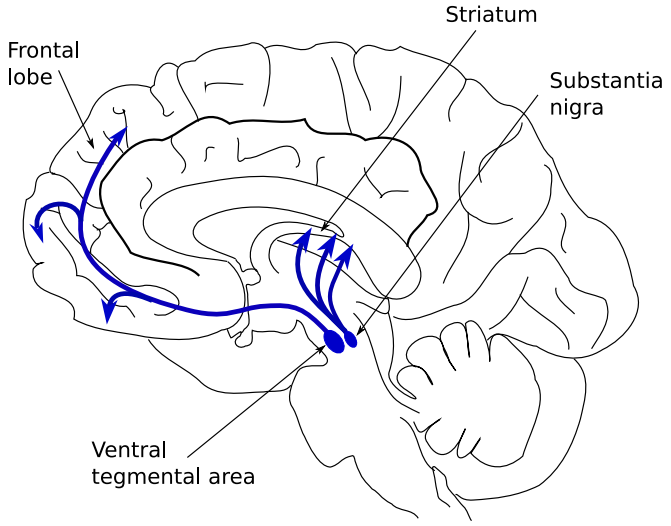
Dopamine is a chemical that is released by special neurons when we receive a reward, e.g. achieving an objective or reaching a goal, playing a game, eating something we like, etc.

Dopaminergic circuits gets activated by reward and pleasure, and regulate motivation. Most addictive drugs affect dopaminergic activity, e.g. increasing the concentration of dopamine in the synaptic cleft by blocking the reuptake (cocaine).

Death of dopaminergic neurons causes Parkinson's disease.

Altered levels of dopaminergic activity, or imbalance of dopaminergic receptor is strongly linked to schizophrenia.



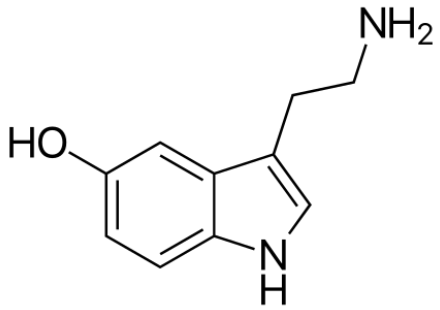


dopaminergic system

Interesting facts - Other neurotransmitters

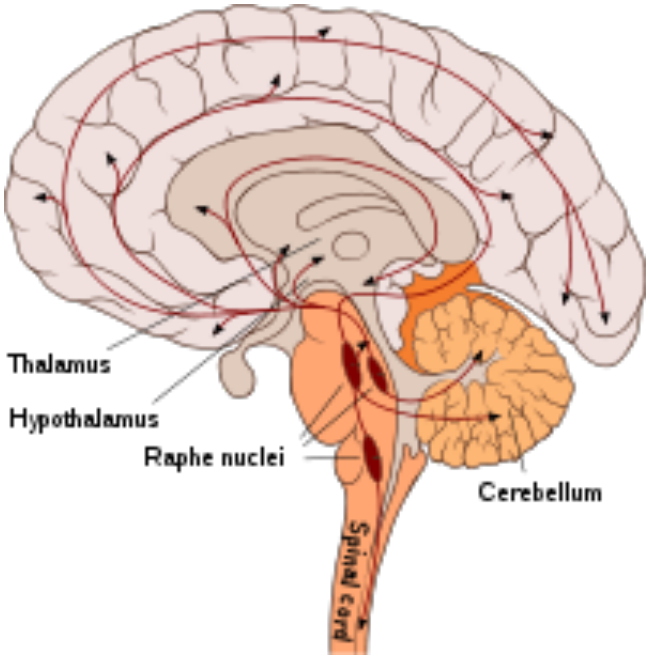
Serotonin is important to regulate mood: low levels of serotonin are associated with depression.

"the serotonin receptor have diverse effects on mood, anxiety, sleep, appetite, temperature, eating behaviour, sexual behaviour, movements and gastrointestinal motility"
(https://en.wikipedia.org/wiki/Serotonin)



In fact, a category of medicines called 'selective serotonin reuptake inhibitors (SSRI) are a type of antidepressant. The work by slowing down the reuptake of serotonin, which then remains in the synaptic cleft longer.

Anandamide. It was also found that neurons have mysterious receptors called endocannabinoids receptors, but the corresponding transmitters are mostly unknown, except from cannabis and the endogenous anandamide.



From brain to computational neural networks

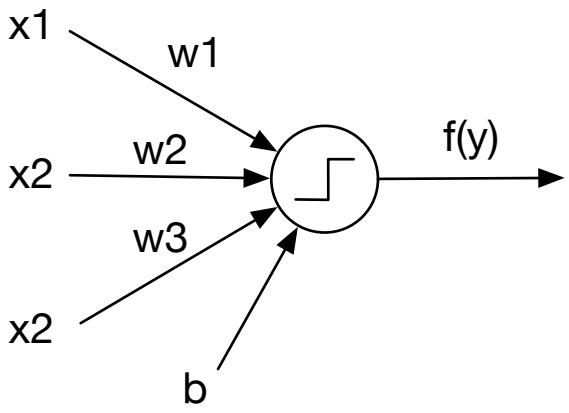
Main message: be aware of the the gap

Neurons and brain structures are considerably more complex than computational neural networks, thus we talk of a “weak inspiration”.

The way neural networks work might have little in common with brain function, and vice versa.

If by ‘artificial intelligence’ we mean neural networks, and by ‘natural/biological intelligence’ we mean what is generated by a brain, we must acknowledge the large gap between the two.

The perceptron



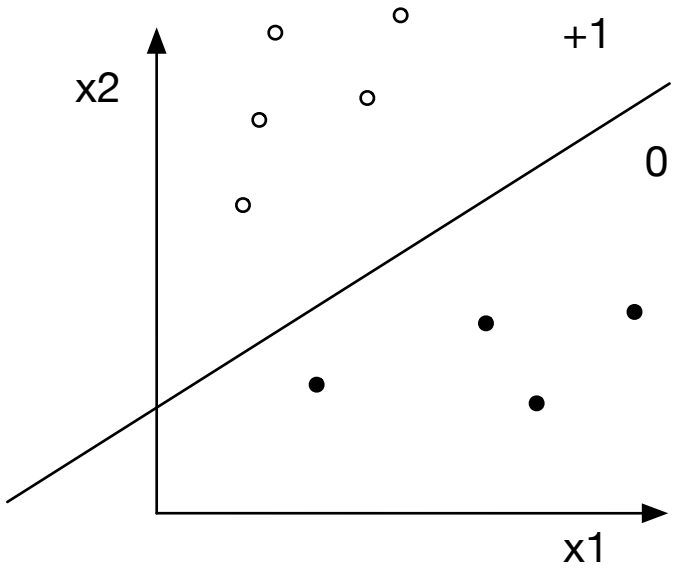
$$f(y) = 1 \quad if \quad \sum (w_i \cdot x_i) + b \geq 0$$

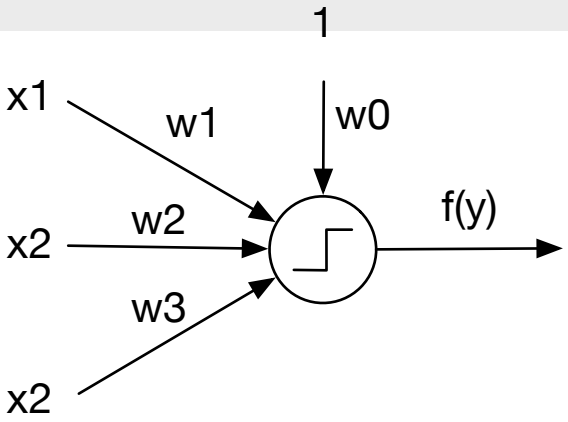
$$f(y) = 0 \quad if \quad \sum (w_i \cdot x_i) + b < 0$$

x are the inputs
w are the weights
b is the bias

The perceptron - II

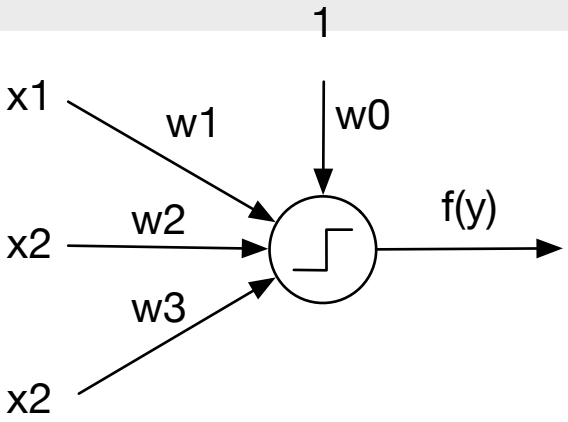
Example with two dimensions:
$f(y) = w_0 + x_1 w_1 + x_2 w_2$





The separating line is expressed by $w_0 + x_1 w_1 + x_2 w_2 = 0$
Note that the bias b can be replaced with an input 1 via the weight w_0

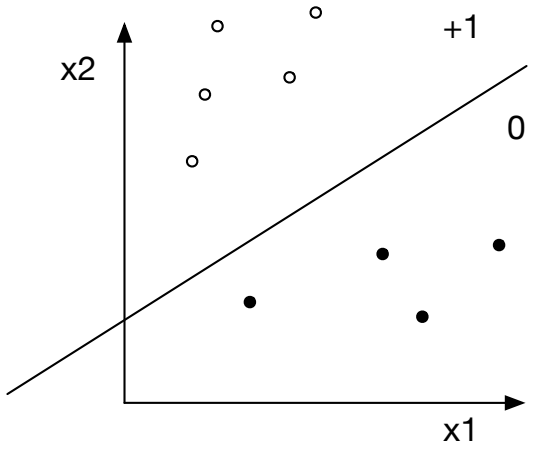
The perceptron - learning rule



Let's consider a set of datapoints:

1. $(x_1(1),x_2(1))$ -> t(1), e.g. class 0
2. $(x_1(2),x_2(2))$ -> t(2), e.g. class 0
3. $(x_1(3),x_2(3))$ -> t(3), e.g. class 1
4. $(x_1(4),x_2(4))$ -> t(4), e.g. class 1

The 'class' is the desired output or target t.
We start by picking random values for w_0, w_1, and w_2



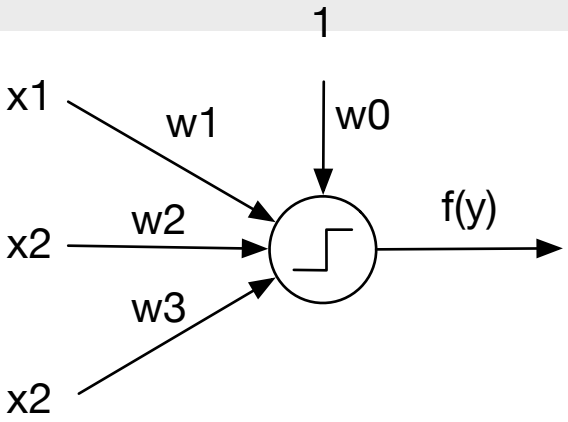
Then we compute $f(y(1)) = H(w_0 + x_1(1)*w_1 + x_2(1)*w_2)$, H is the step function. If f(y(1)) = t(1) then the prediction is correct. If not, we update the weights according to the equations:

$w_0 = w_0 + n(t(1) - f(y))$
$w_1 = w_1 + n(t(1) - f(y))x_1(1)$
$w_2 = w_2 + n(t(1) - f(y))x_2(1)$

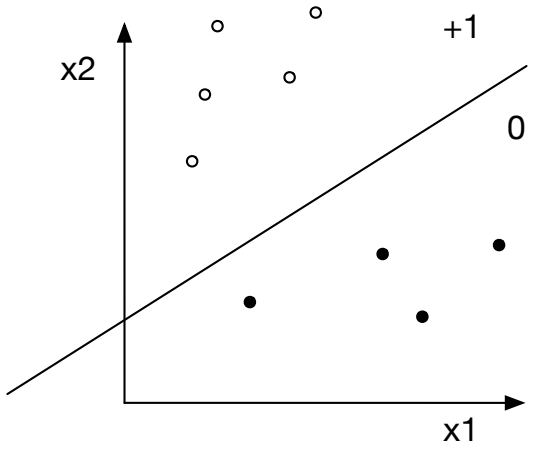
Note that n is a 'learning rate'.

Main idea in ML: learning a model from the data

Rather than building a complex model before hand, ML attempts to acquire knowledge and include that knowledge into a model, which in this case is represented by the transfer function, i.e. the step function, and the weights w0,w1,w2 and w3.

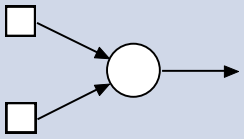
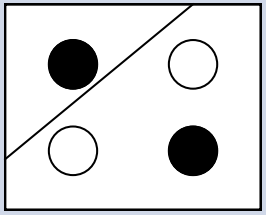
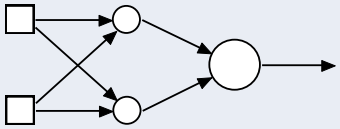
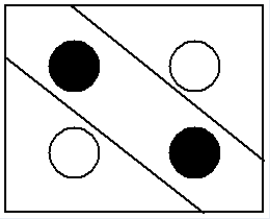
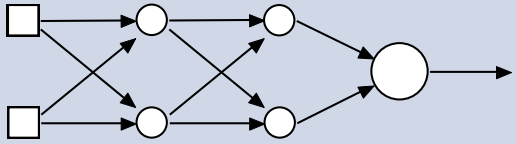
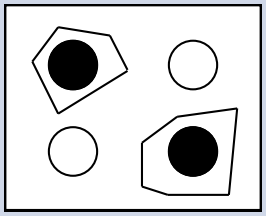


In this way, different models can be learned for different types of data or data distribution.



The perceptron - limitations

One perceptron alone cannot solve the XOR problem

Structure	Layers	XOR Problem
	Single	
	Two	
	Three	

The perceptron - limitations

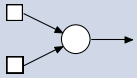
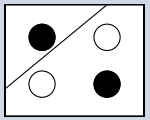
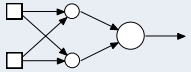
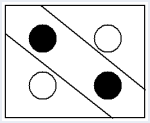
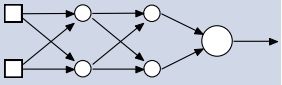
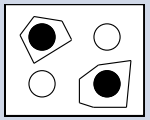
Multi layer perceptrons can solve more complex problems

BUT

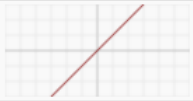
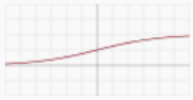
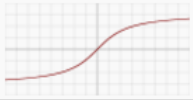
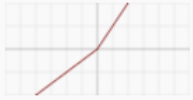
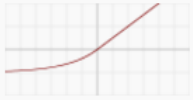
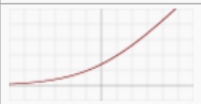
how to train the weights w?
No method exists.

Why? Because we know the error at the output, and we can change the weights of the last layer, but what about the previous layers?

This problem caused a decline in the belief that neural networks could be useful in the 60' and 70'.

Structure	Layers	XOR Problem
	Single	
	Two	
	Three	

Activation functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

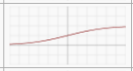
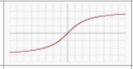
Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, accessed 11/2/2019

Activation functions: with derivative vs. without derivative

If the activation function has a derivative, it is possible to compute the derivative of the error with respect to each parameter of the model, and therefore, understand how the output changes in relation to changes of the parameters.

From this idea, the concept of backpropagation of error was derived.

Before looking at backpropagation, let's introduce the problem of image classification and the concept of loss.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, accessed 11/2/2019

Image classification

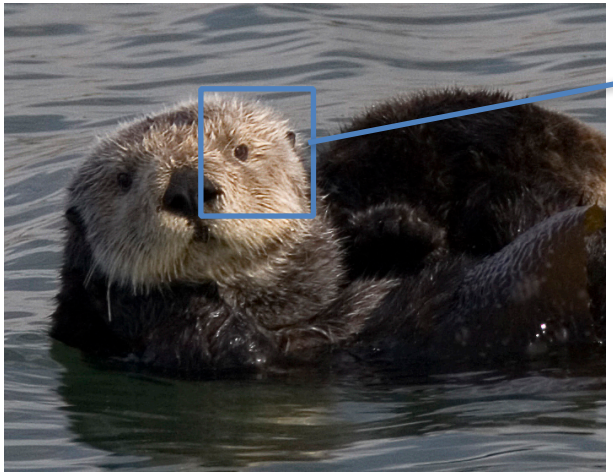
Assume that we have a set of labels, e.g. cat, dog, otter, koalas

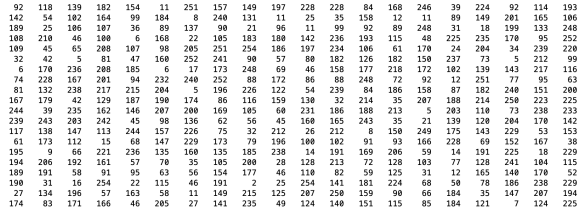
otter

Note: the following material in this lecture is adapted from CS321n (Stanford University), in particular from the lectures of Andrei Karpathy, Justin Johnson and Serena Yeung.

Image classification: the problem

Assume that we have a set of labels, e.g. cat, dog, otter, koalas



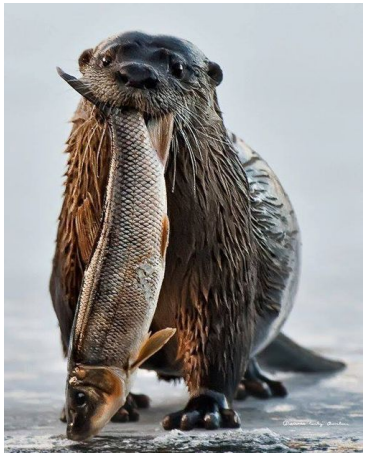


otter

The computer sees a large matrix of numbers. It's not easy to create a function that performs a computation on these numbers and gives 'otter' as an answer.

Image classification: the problem

- different camera angles / positions
- different illumination
- different subtypes (e.g. young/adult)
- deformation
- partially visible
- background clutter/other objects



Example: linear classifier

image, e.g. 32x32x3,
total 3072 numbers

f(x,W)

W are the parameters or weights

10 numbers giving class scored, e.g. for 10 different types of animals

Example: linear classifier

image, e.g. 32x32x3,
total 3072 numbers

$f(x,W) = Wx + b$

W are the parameters or weights

10 numbers giving class scored, e.g. for 10 different types of animals

Example: linear classifier

W = 10x3073 x = 3073x1 f = 10x1

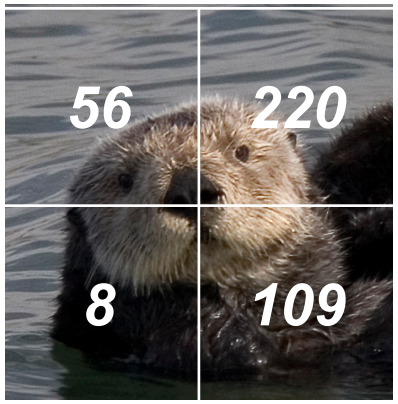
image, e.g. 32x32x3,
total 3072 numbers

$f(x,W) = Wx + b$

10 numbers giving class scored, e.g. for 10 different types of animals

W are the parameters or weights

Example: linear classifier



0.2	-0.5	0.1	2.0
1.5	1.5	2.1	0.0
0.0	0.25	0.2	-0.3

W

3x4

56
220
8
109

x

4x1

+

6
-20
4

b

=

126	otter score
411	cat score
28	dog score

3x1

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

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

the other classes scores

the true class score

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

Loss: **3**

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

Loss: **3**

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

car loss: max(0, 5 - 6 + 1) + max(0, 3 - 6 + 1)
= max(0,0) + max(0,-2) = 0

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

Loss: **3** **0**

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

car loss: max(0, 5 - 6 + 1) + max(0, 3 - 6 + 1)
= max(0,0) + max(0,-2) = 0

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***

Loss: **3** **0**

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

car loss: max(0, 5 - 6 + 1) + max(0, 3 - 6 + 1)
= max(0,0) + max(0,-2) = 0

airplane loss: max(0, 3 - (-2) + 1) + max(0, 4 - (-2) + 1)
= max(0,6) + max(0,7) = 13

Example: scores and loss function

			
airplane	***-2.0***	***5***	***-2***
automobile	***5***	***6***	***3***
cat	***3***	***3***	***4***
Loss:	**3**	**0**	**13**

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

the other classes scores

the true class score

cat loss: max(0, 5 - 3 + 1) + max(0, -2 - 3 + 1)
= max(0,3) + max(0,-4) = 3

car loss: max(0, 5 - 6 + 1) + max(0, 3 - 6 + 1)
= max(0,0) + max(0,-2) = 0

airplane loss: max(0, 3 - (-2) + 1) + max(0, 4 - (-2) + 1)
= max(0,6) + max(0,7) = 13

Main types of loss function

From the previous lecture on polynomial curve fitting, remember:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 \qquad (1.2)$$

A measure of error is also called ‘loss’.

In the case of classification, e.g. of images, a loss function that tells us how wrong the classifier is.

There are a variety of loss functions, but the general idea is to capture a measure of **how wrong the classifier is in respect to the ground truth**.

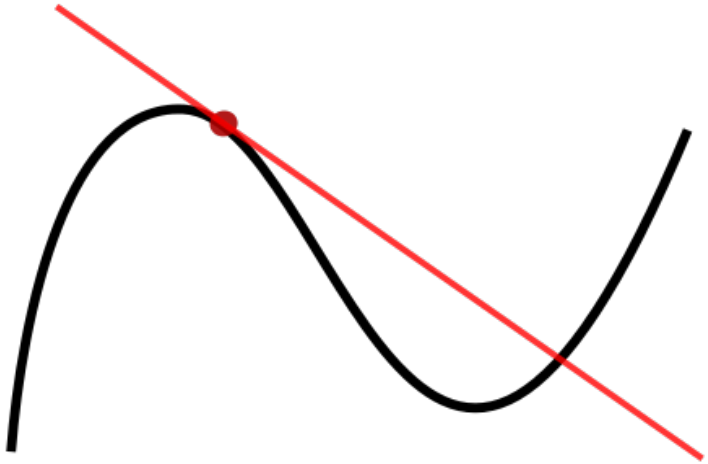
A way to train multi layers neural networks

Werbos (1982), Widrow (1990) Backpropagation
Rumelhart, Hinton, Williams., (1986): Learning representations by back-propagating errors.
Also see:
http://people.idsia.ch/~juergen/who-invented-backpropagation.html

Main idea:

If, for each node in my network, I can determine the effect on the output when I change one parameter, I can use this information to change parameters in my networks to obtain the desired output.

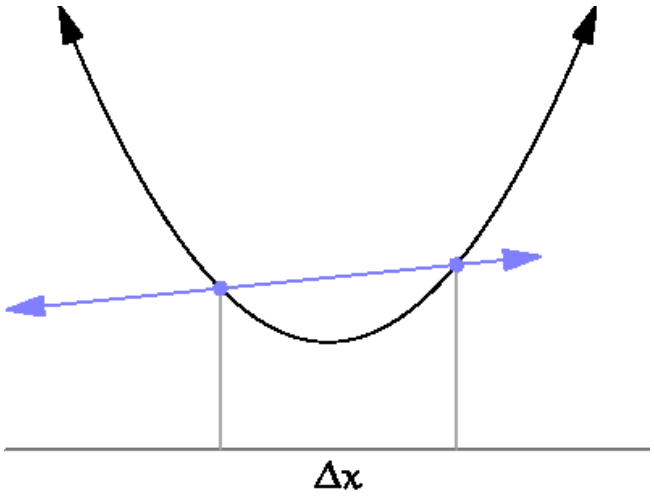
To do so, it is necessary to introduce the concept of **derivative**.



III Derivative

In mathematical terms:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \quad \text{or} \quad f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

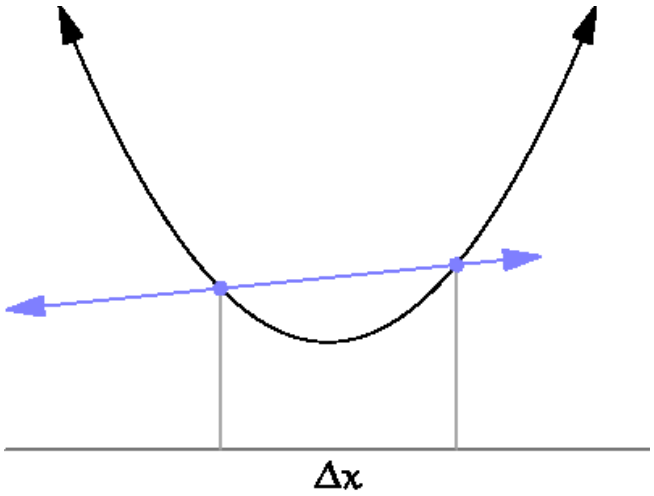


f'(x) is also indicated as df/dx, or d(f)/d(x)

III Derivative

In mathematical terms:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \quad \text{or} \quad f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

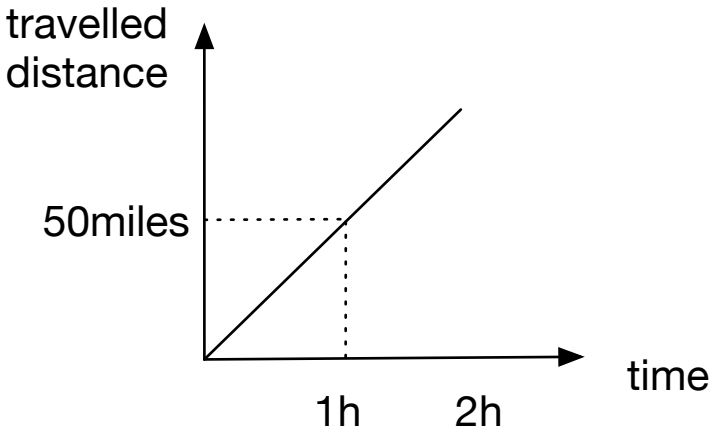


f'(x) is also indicated as df/dx, or d(f)/d(x)

Derivative - example

Assume I'm driving a car.

I could ask the question:
how much does the distance change with time?
I.e. d(km)/d(t)?
Answer: time has a positive effect on the distance.
So if I travel 1h, the distance is 50miles,
if I travel 2h, the distance is 100miles.



Therefore the 'impact' of time on the distance is 50.
50 happens to be the speed, which is the derivative of the distance with respect to time and indicates the steepness of the line in the graph.

Therefore, if I want to go further, I need to travel for longer time.

By changing the time I travel (input), I can achieve the desired effect (output) because I know the effect of time on the travelled distance.

Also note that the rate at which the distance changes is constant, i.e. the speed is constant, which means that the acceleration, the second derivative of the space over time, is zero (0)

Derivative - examples of known derivatives

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

$$f(x) = a \quad \rightarrow \frac{df}{dx} = 0$$

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

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

$$\text{note} : x^{-1} = 1/x$$

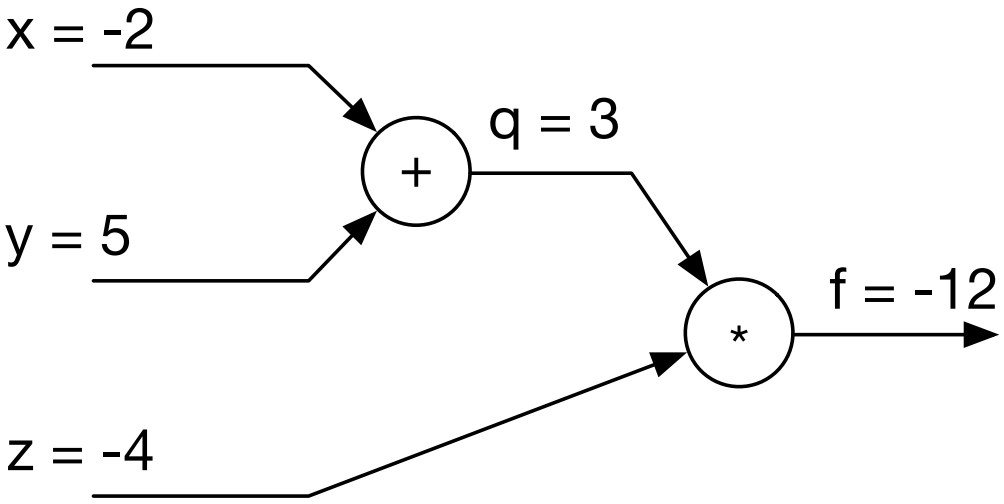
$$f(x) = c + ax \quad \rightarrow \frac{df}{dx} = a$$

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

Use of derivates in computational graphs

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

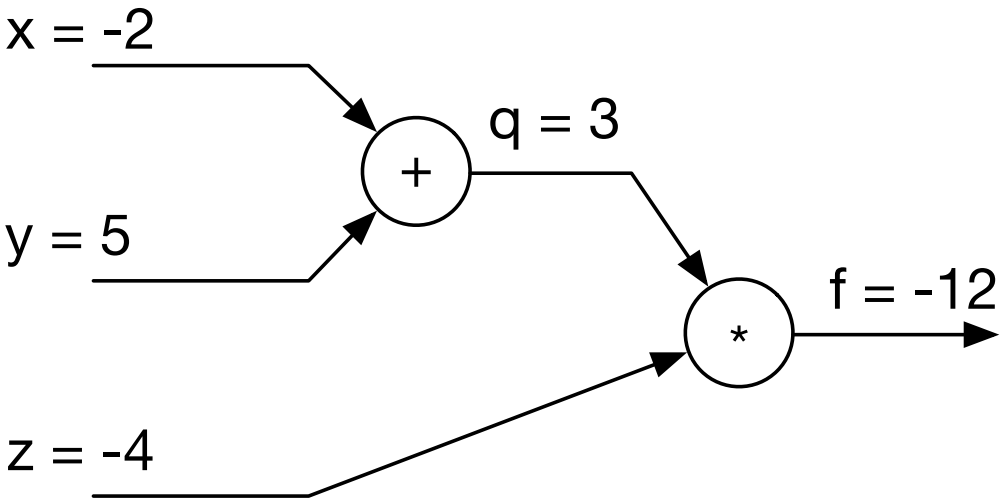
e.g. x = -2, y = 5, z = -4



Use of derivates in computational graphs

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

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

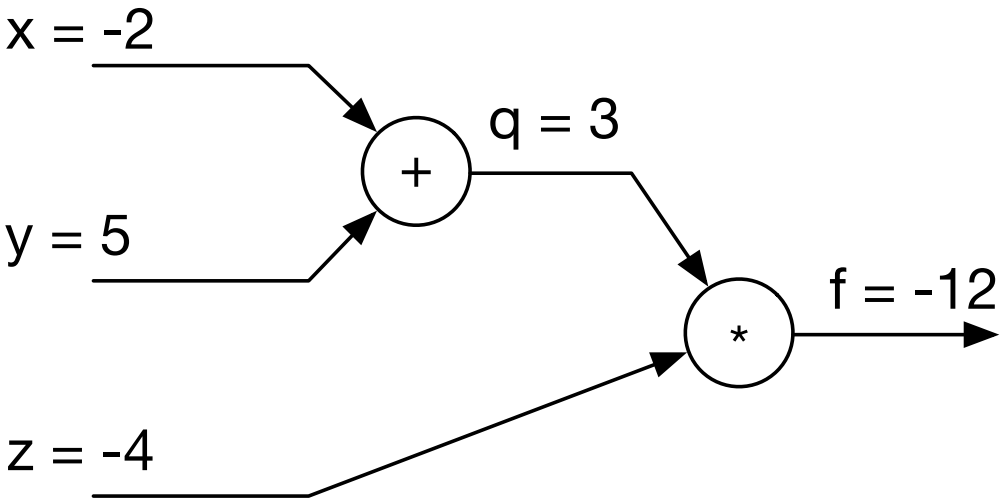


What we want is: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Use of derivates in computational graphs

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

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



What we want is: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

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

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

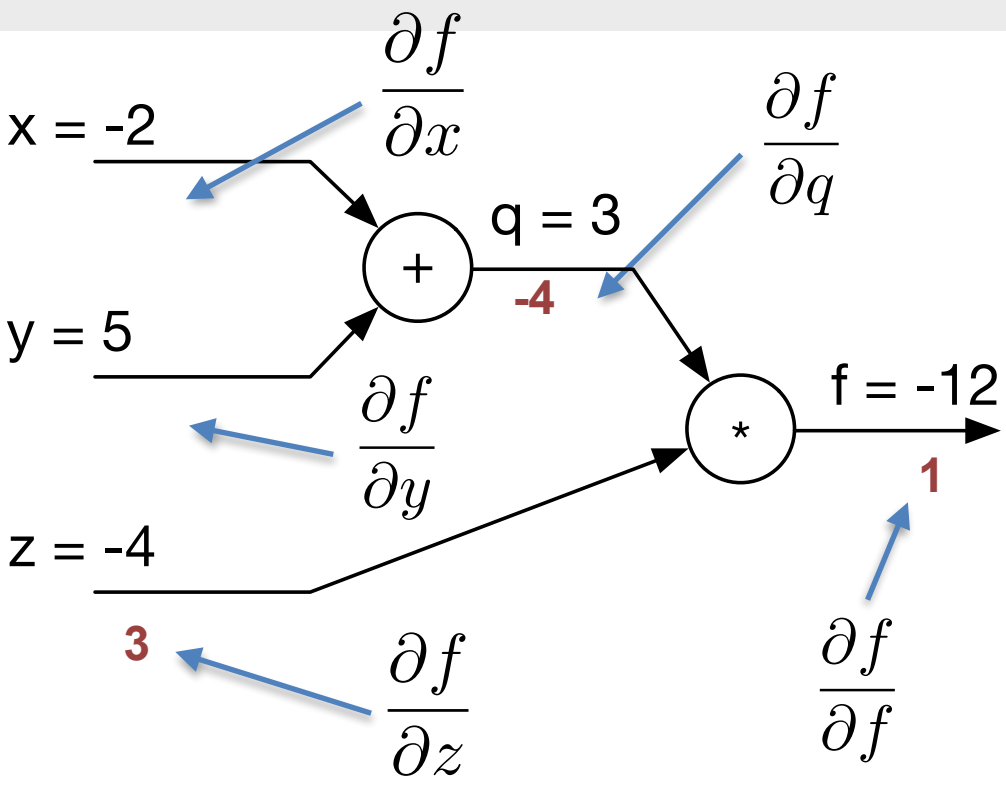
Use of derivates in computational graphs - II

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

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

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

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



if $F(x) = f(g(x))$

then $F'(x) = f'(g(x))g'(x)$

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

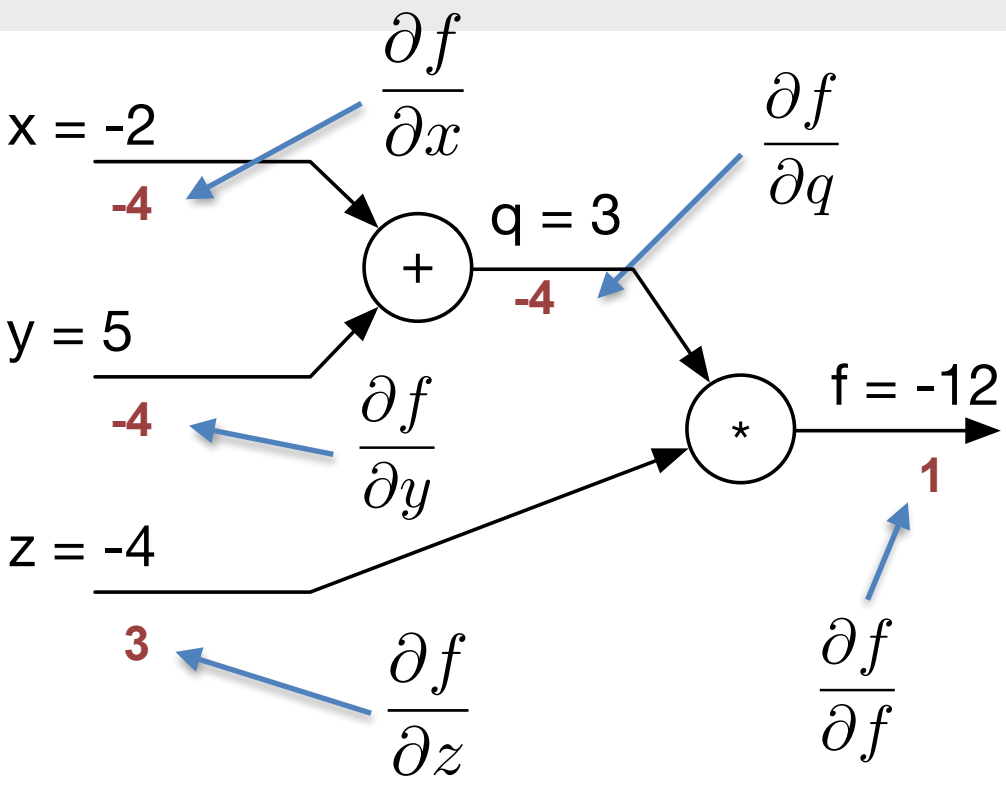
Use of derivates in computational graphs - II

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

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

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

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

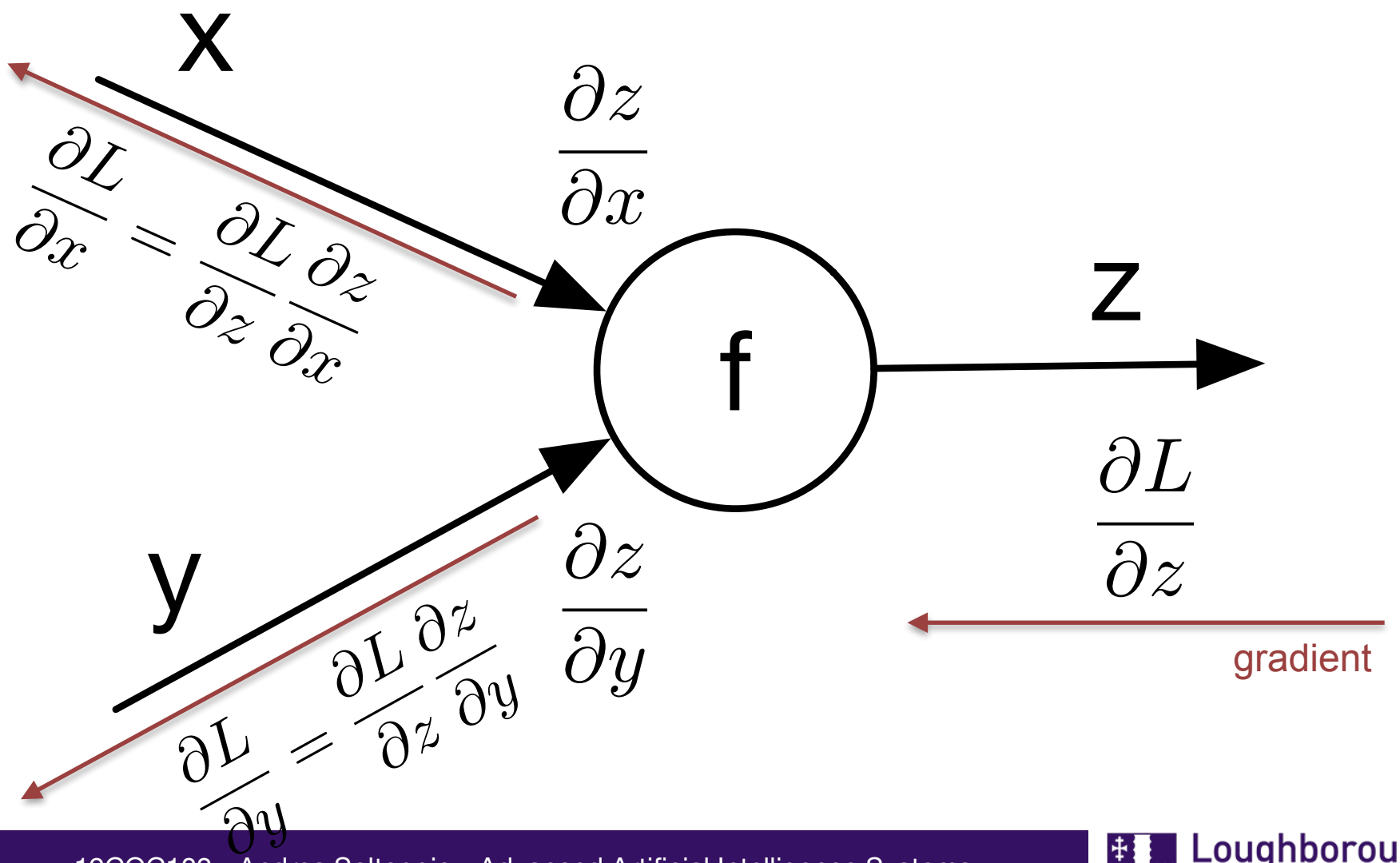


if $F(x) = f(g(x))$

then $F'(x) = f'(g(x))g'(x)$

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

III Use of derivates in computational graphs - III



A second example

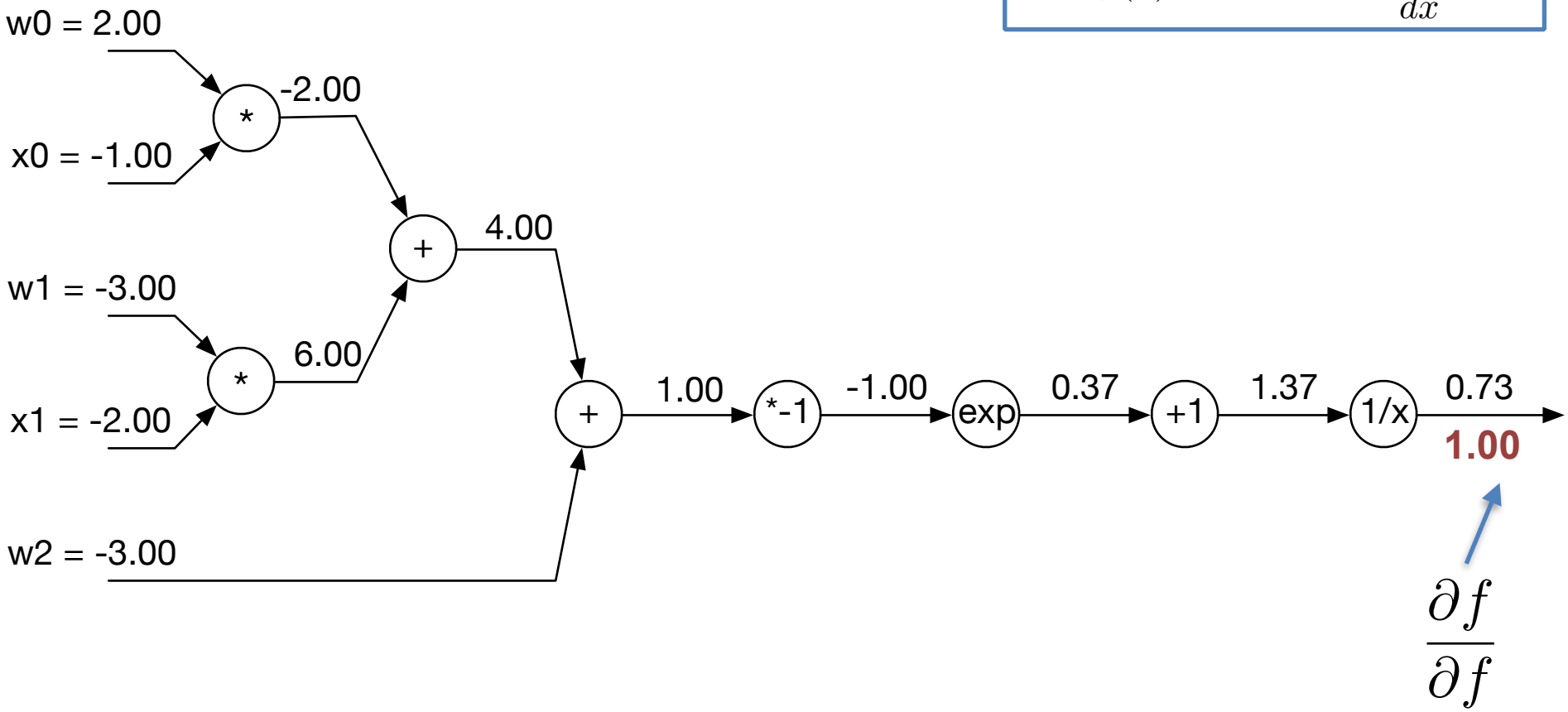
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w2)}}$$

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

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

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

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



A second example

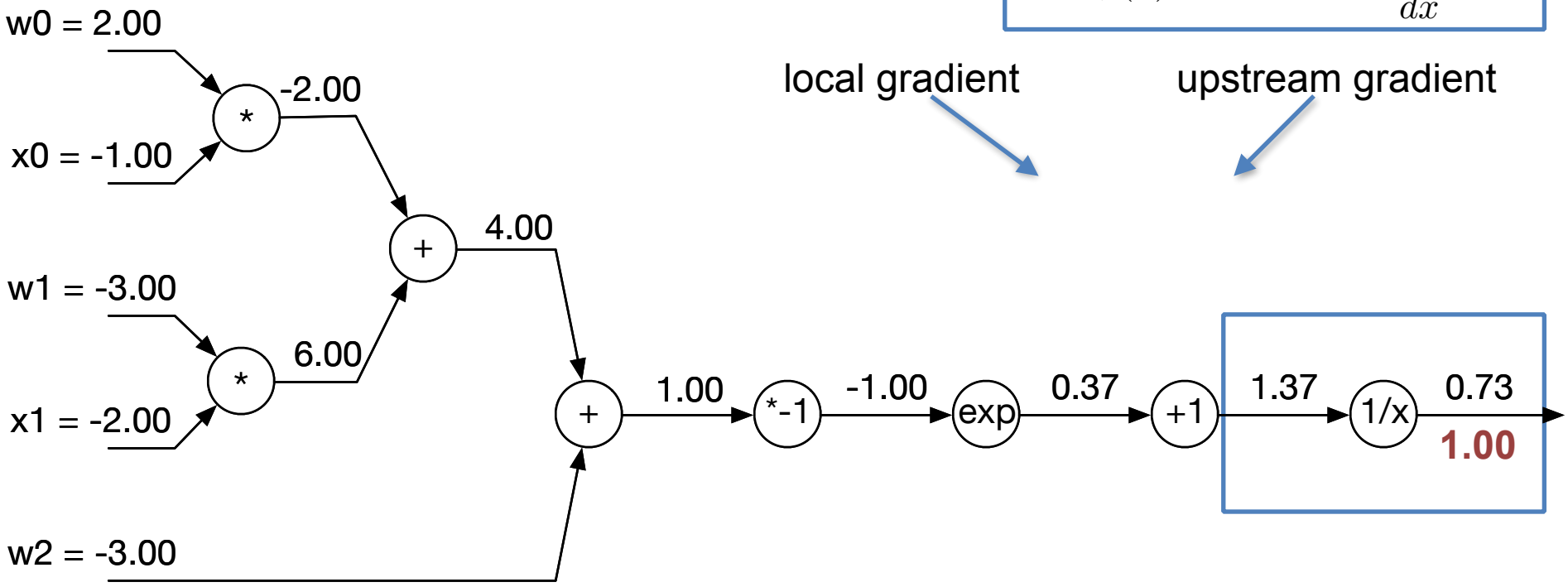
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

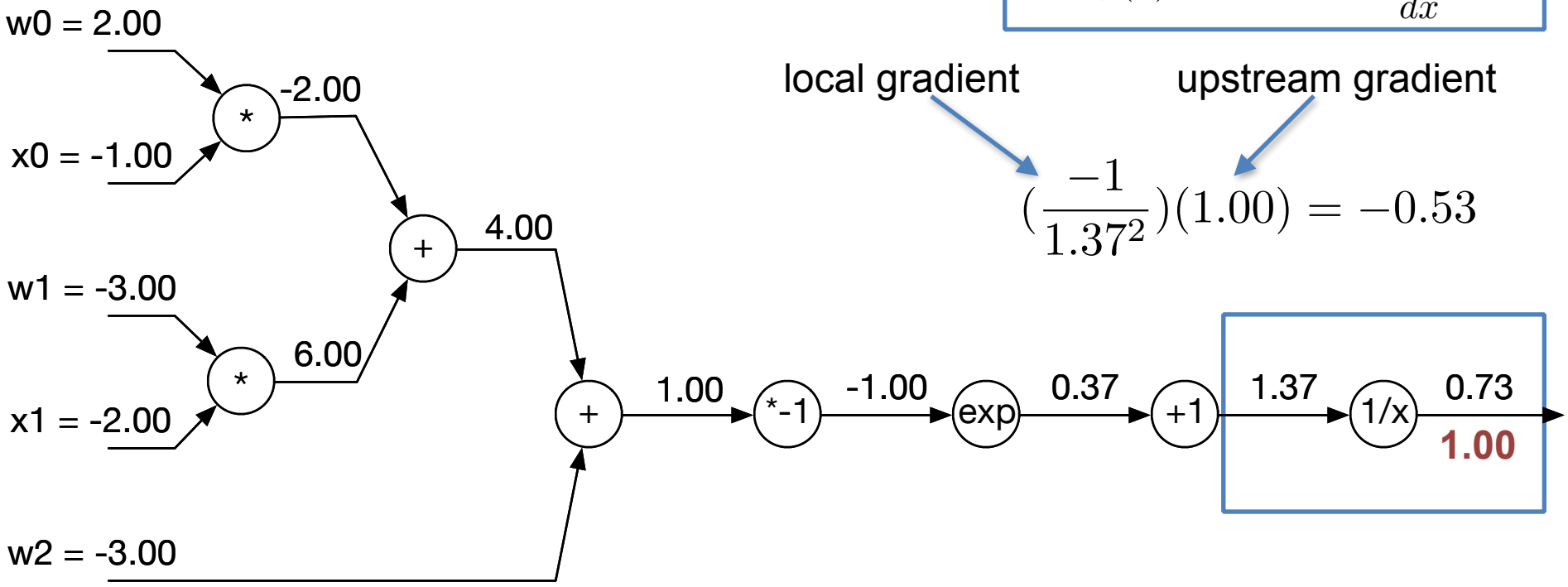
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



local gradient

upstream gradient

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

A second example

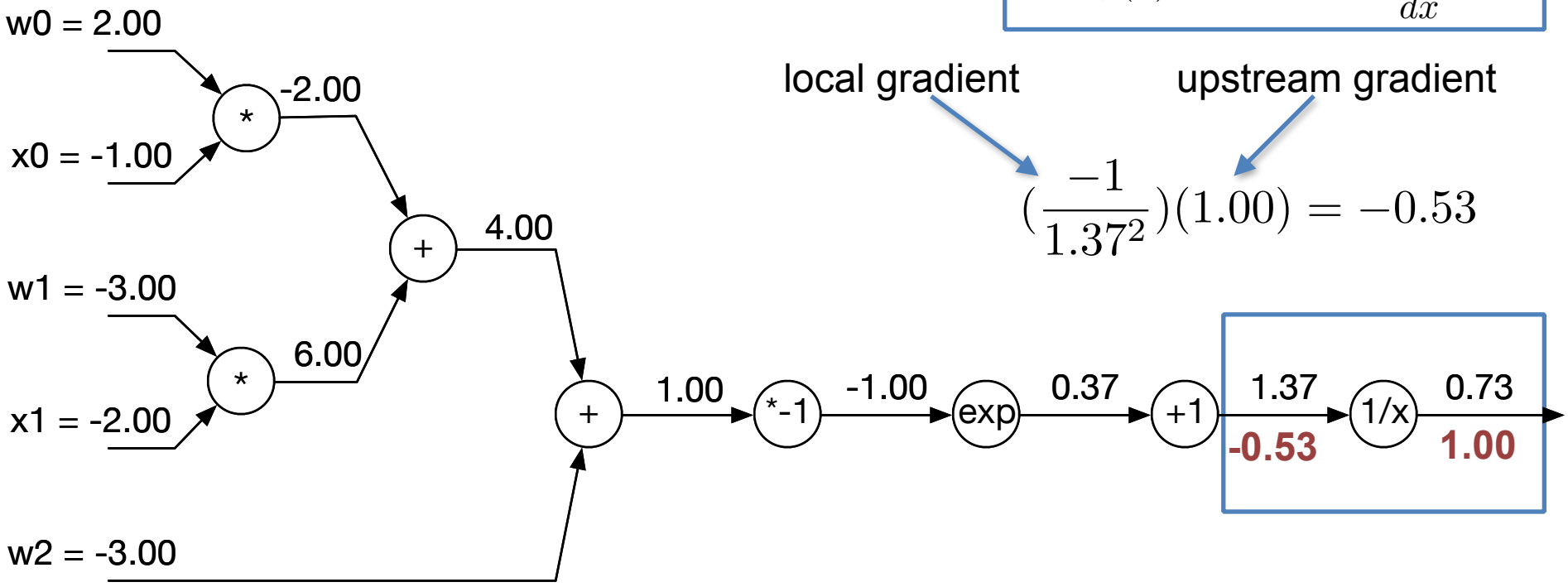
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



local gradient

upstream gradient

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

A second example

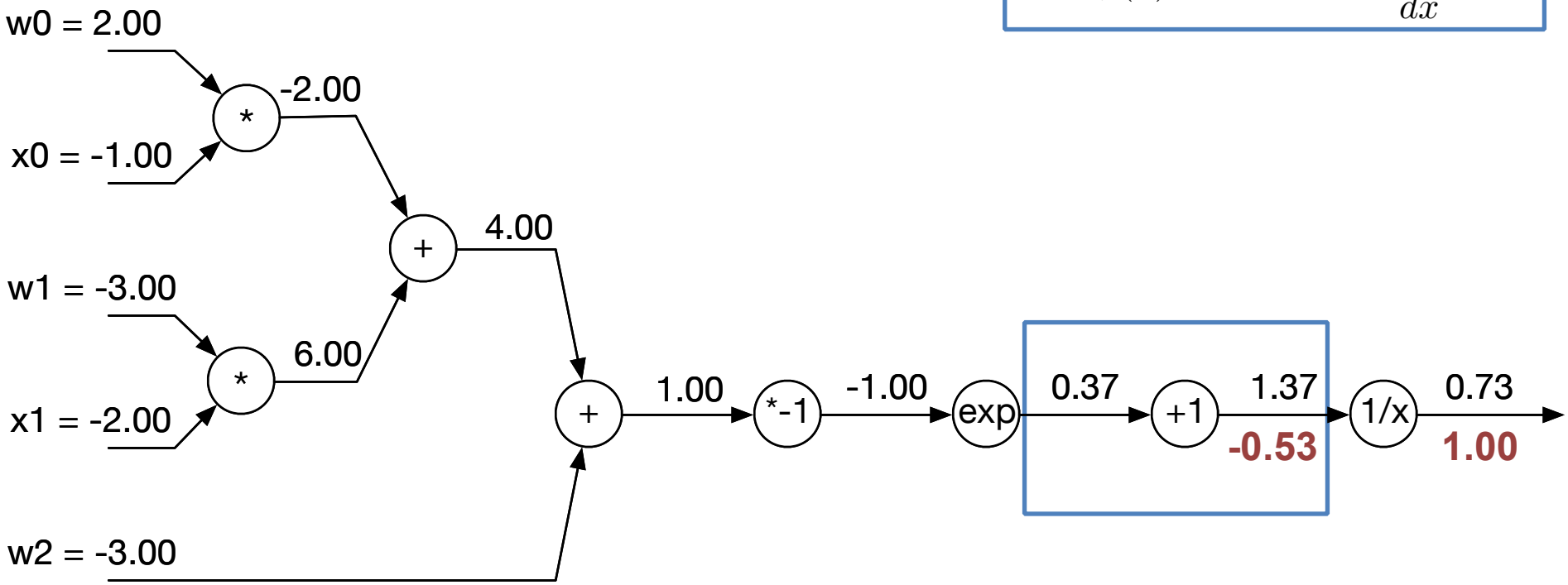
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w2)}}$$

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

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

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

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



A second example

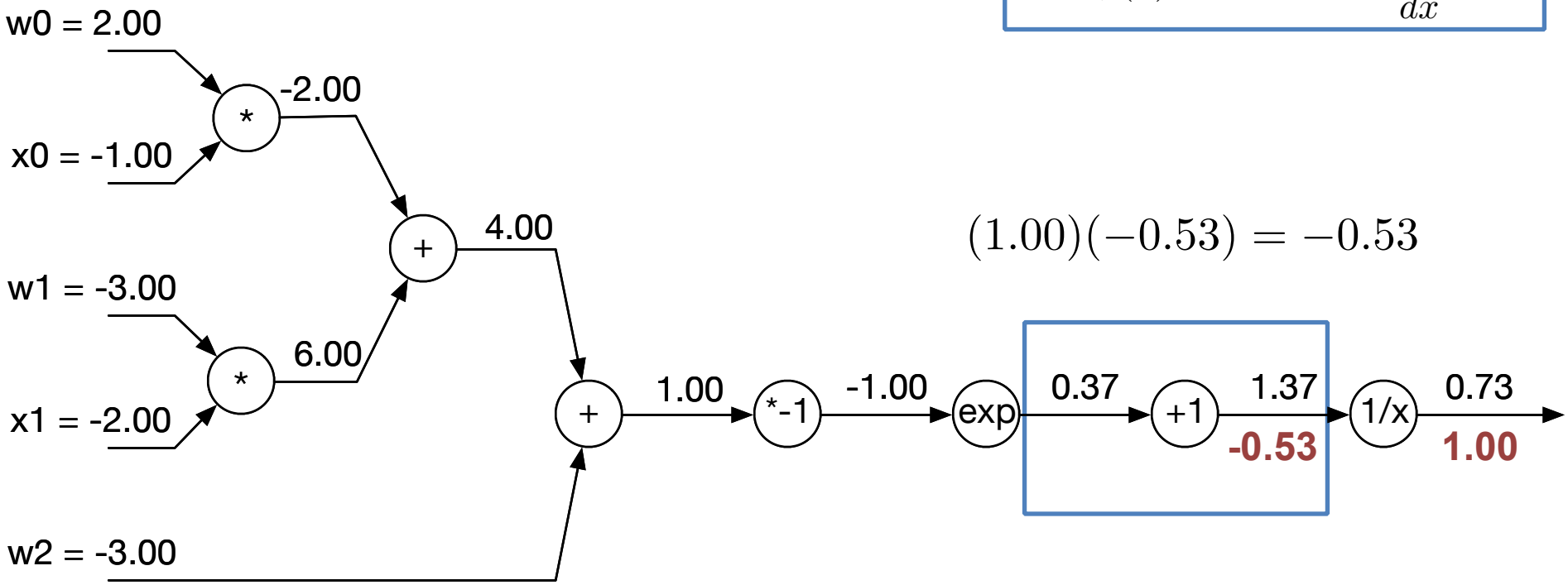
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



$$(1.00)(-0.53) = -0.53$$

A second example

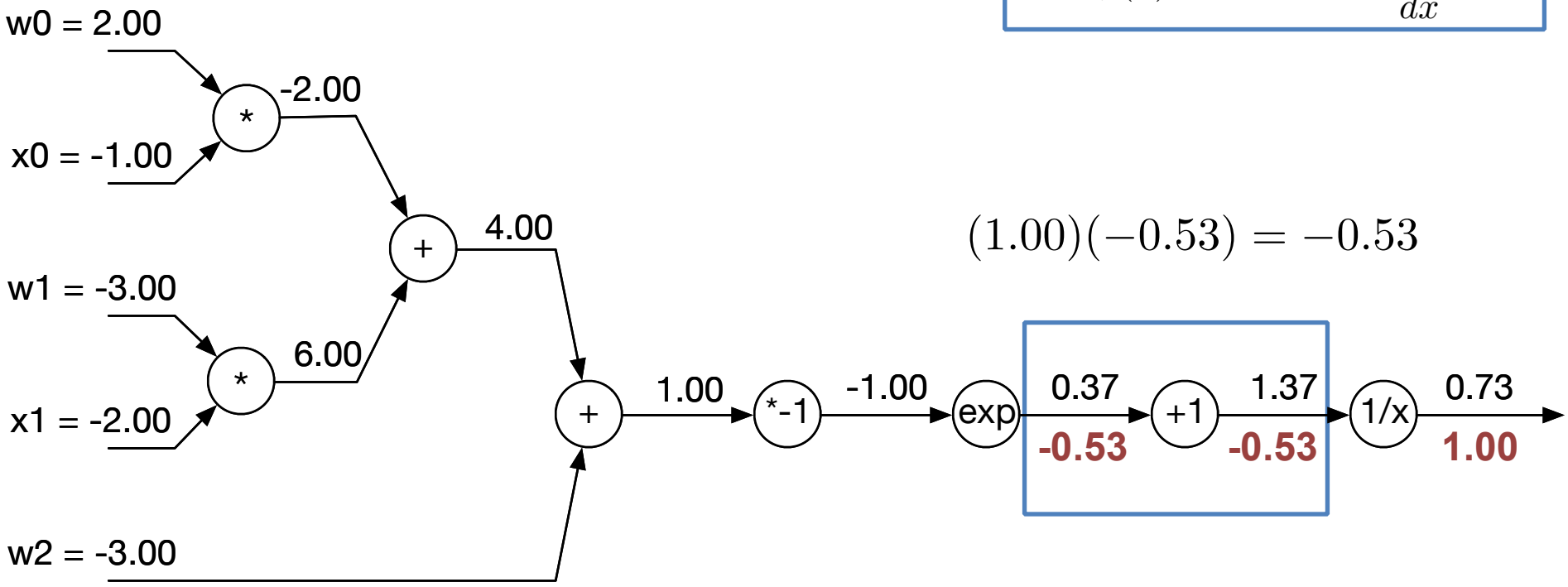
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



$$(1.00)(-0.53) = -0.53$$

A second example

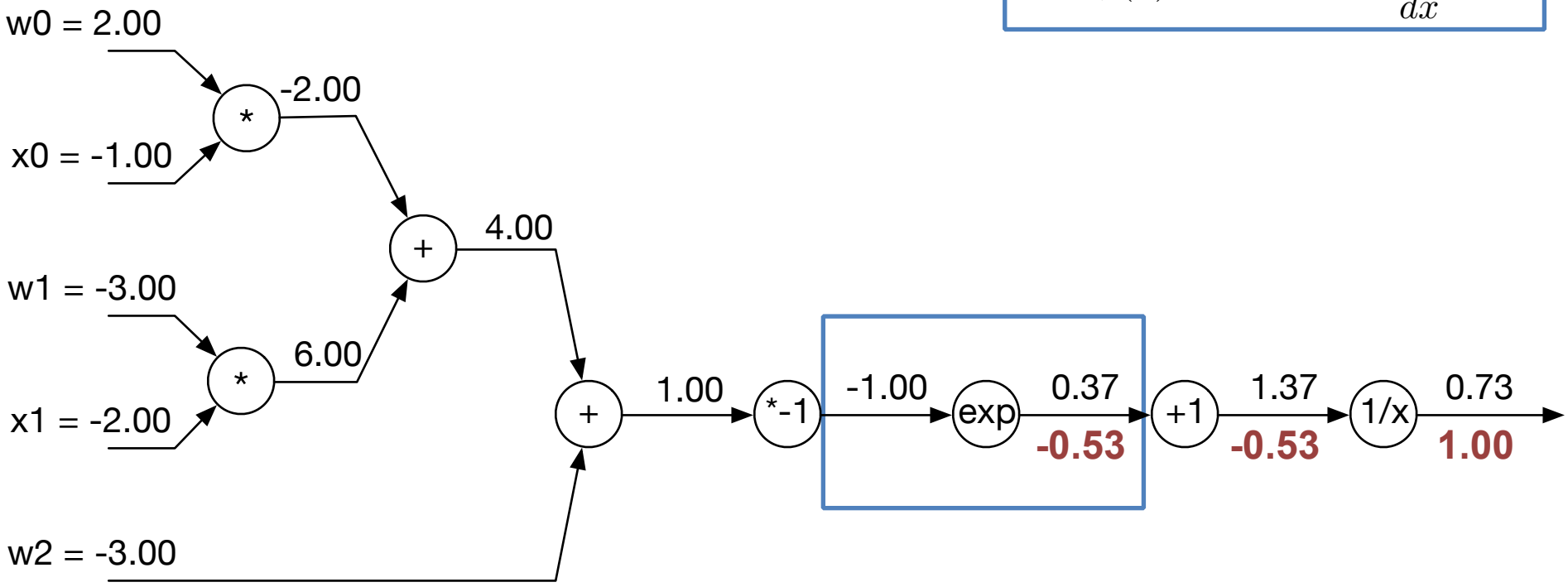
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

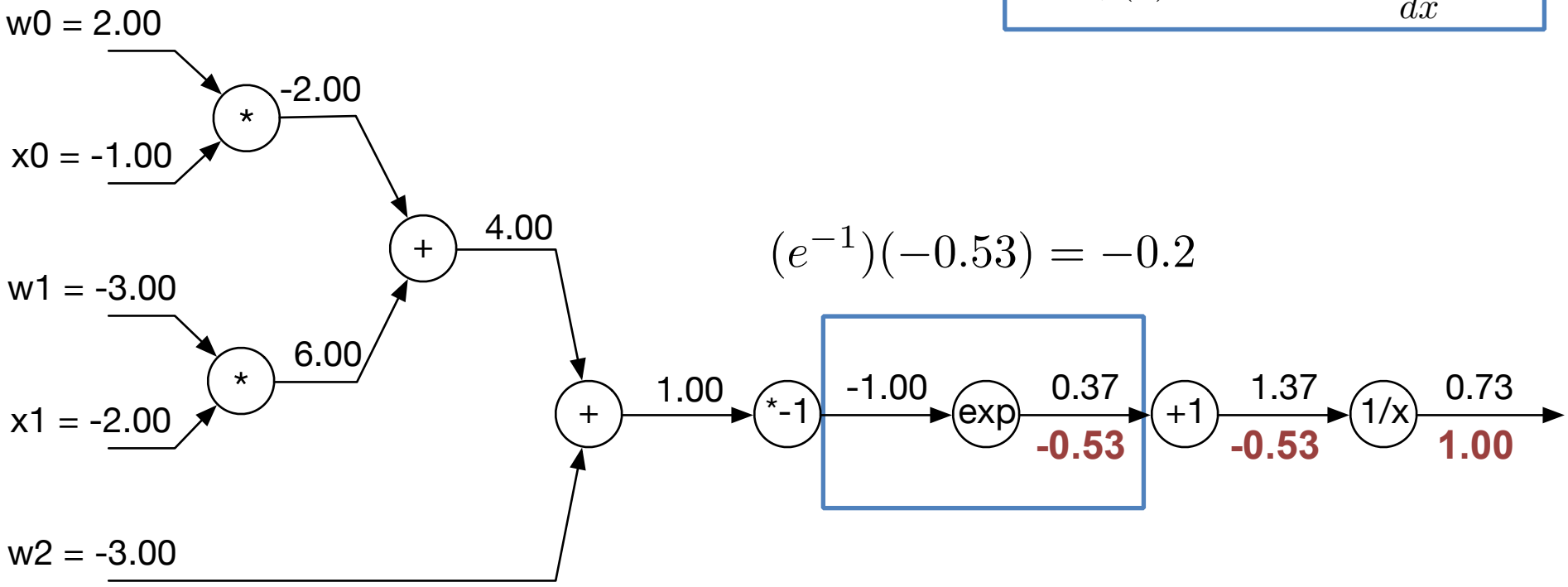
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



$$(e^{-1})(-0.53) = -0.2$$

A second example

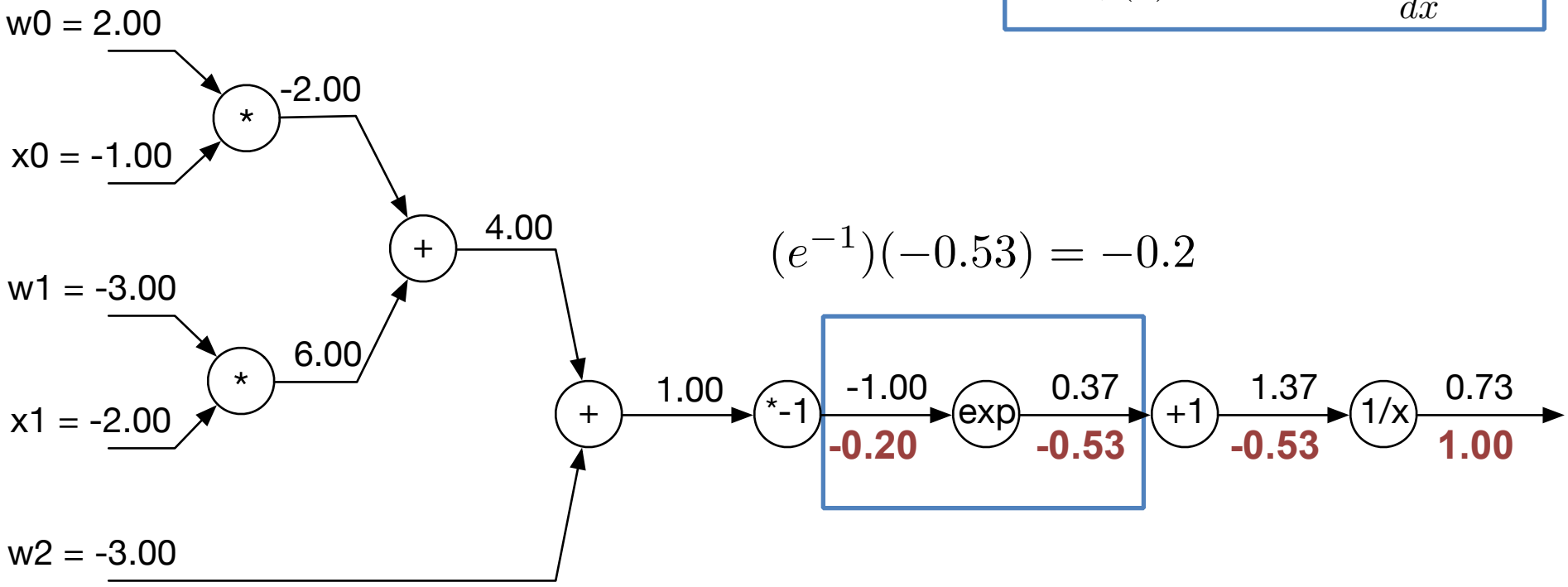
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



$$(e^{-1})(-0.53) = -0.2$$

A second example

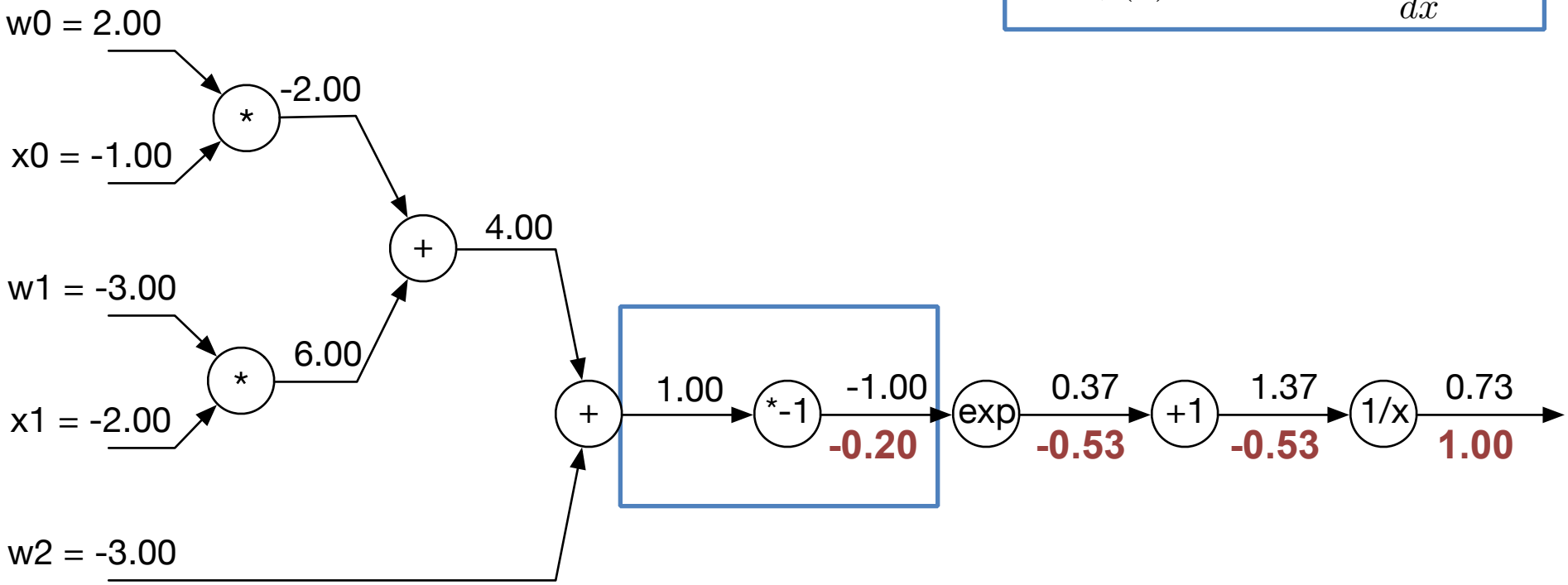
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

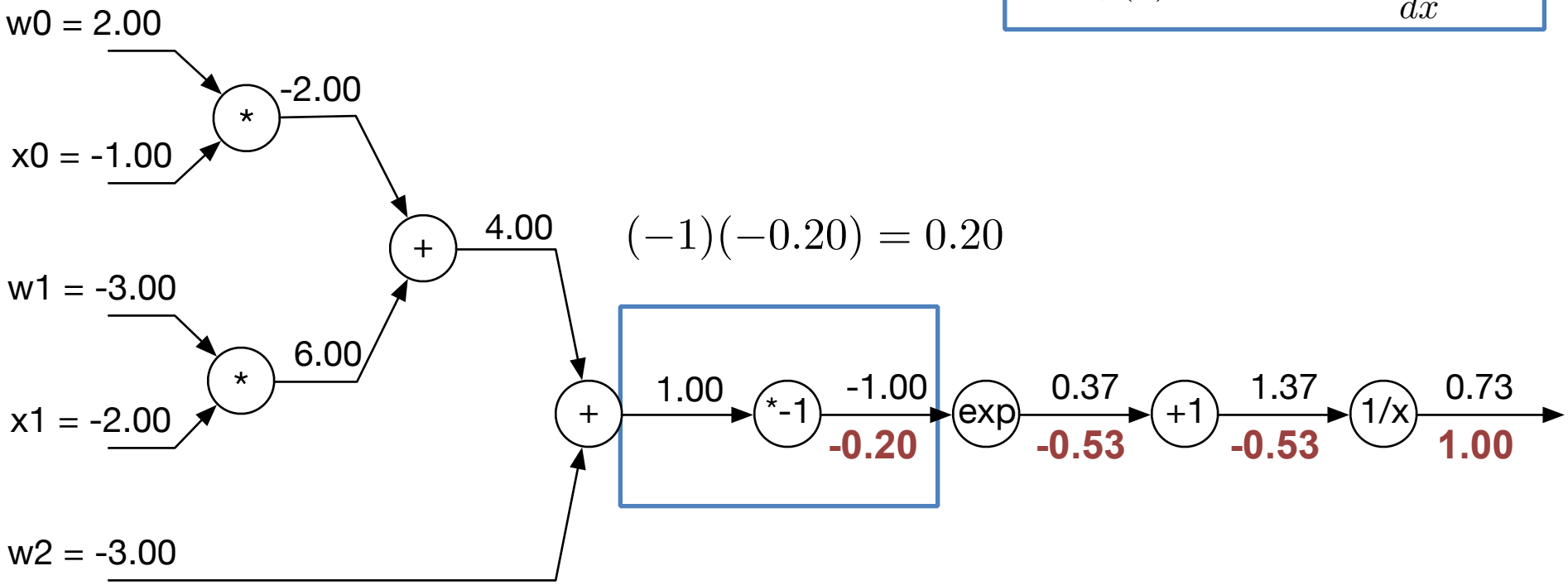
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

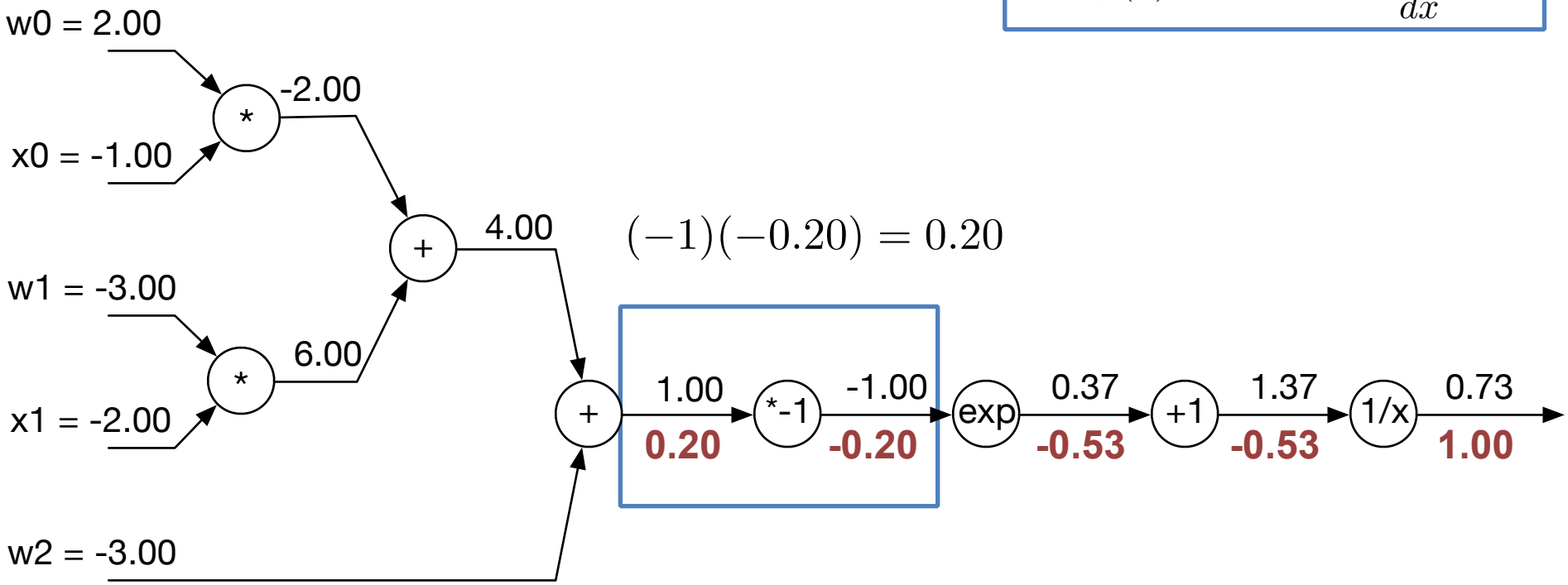
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



$$(-1)(-0.20) = 0.20$$

A second example

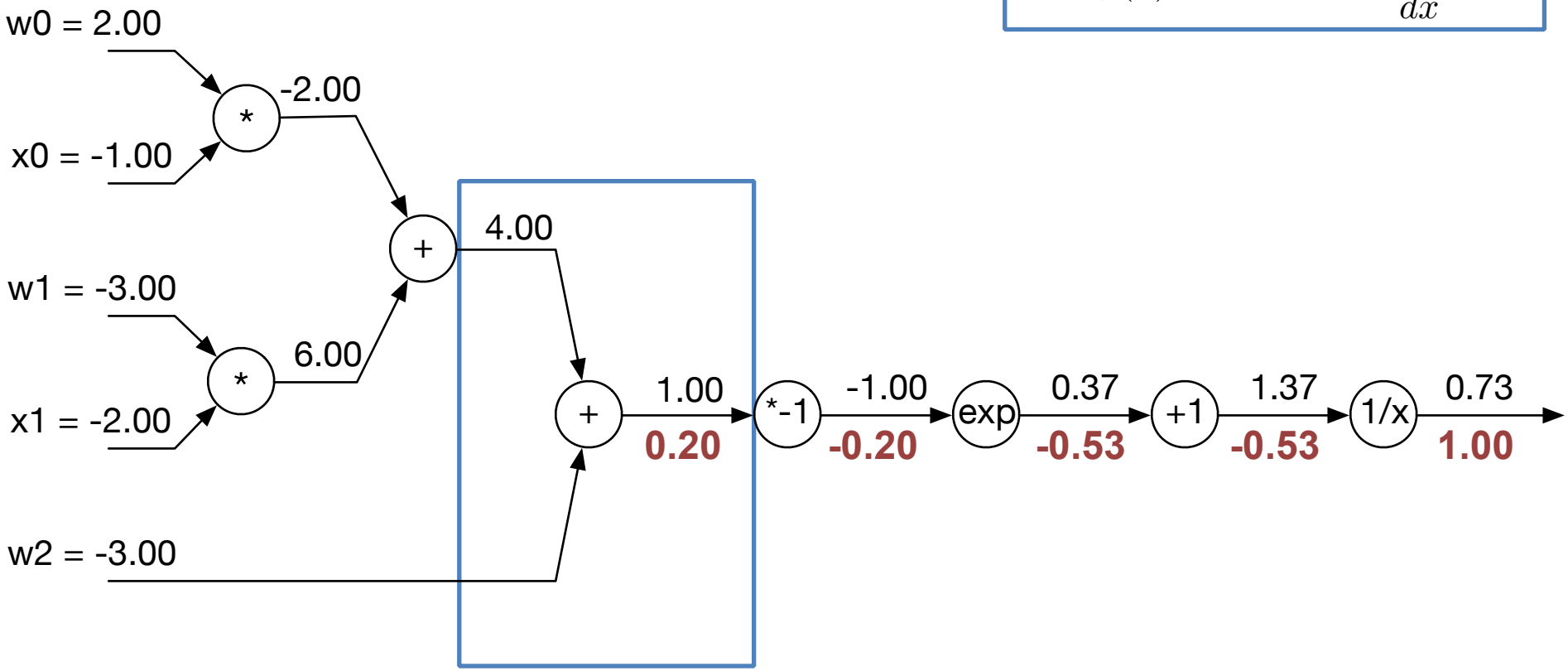
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

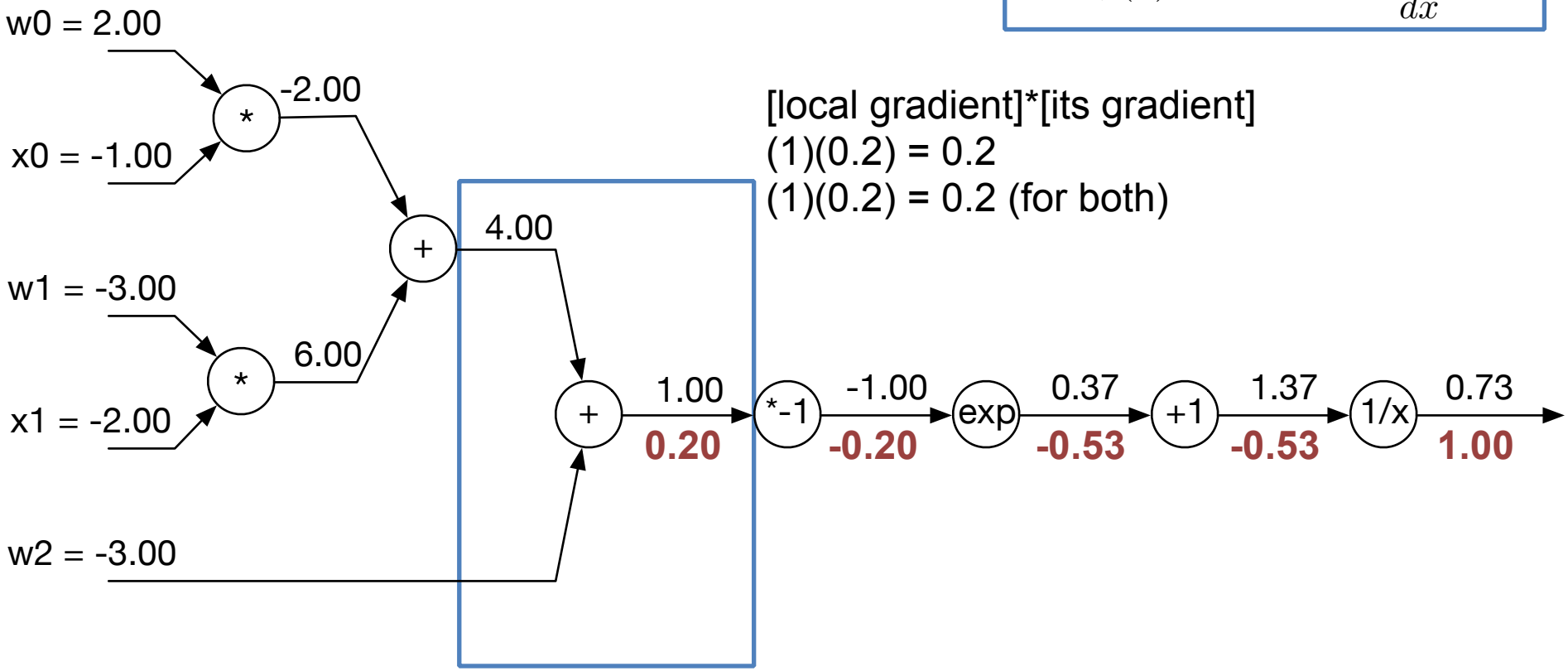
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



[local gradient]*[its gradient]
(1)(0.2) = 0.2
(1)(0.2) = 0.2 (for both)

A second example

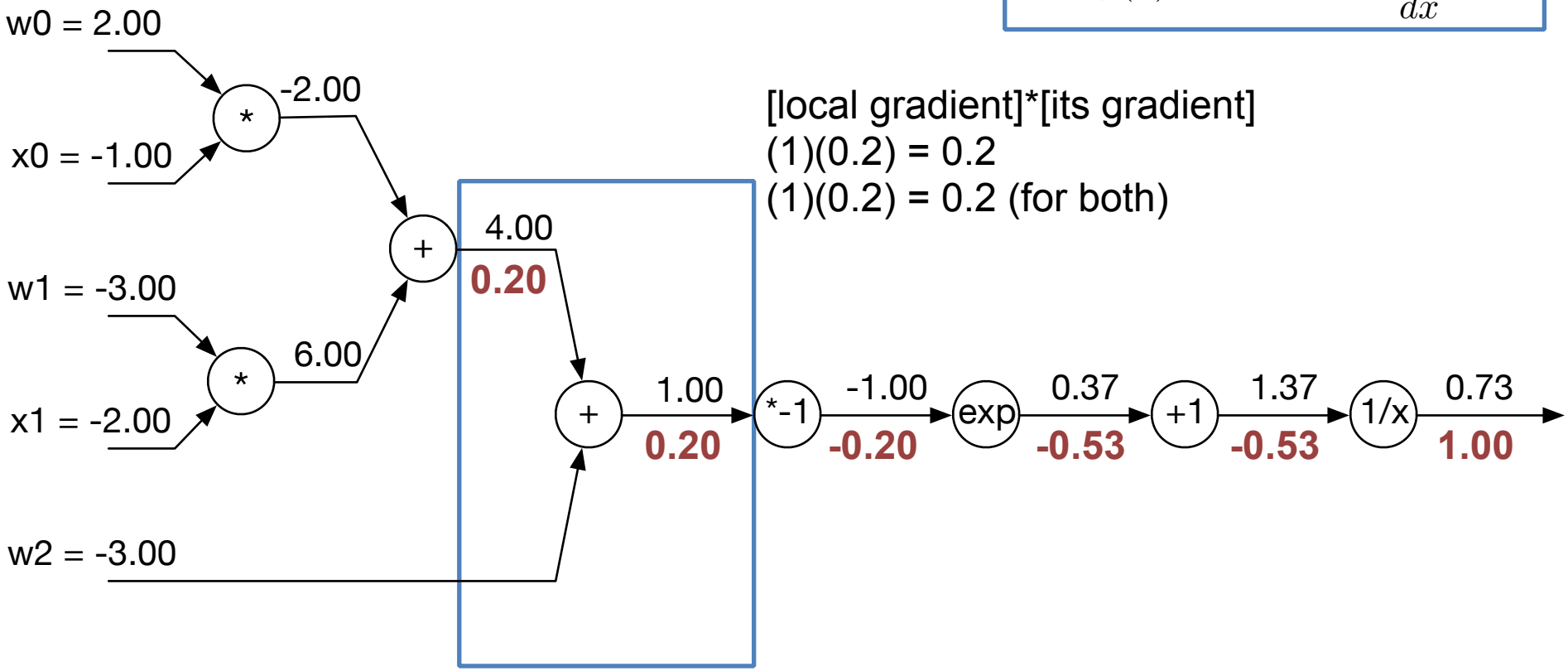
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

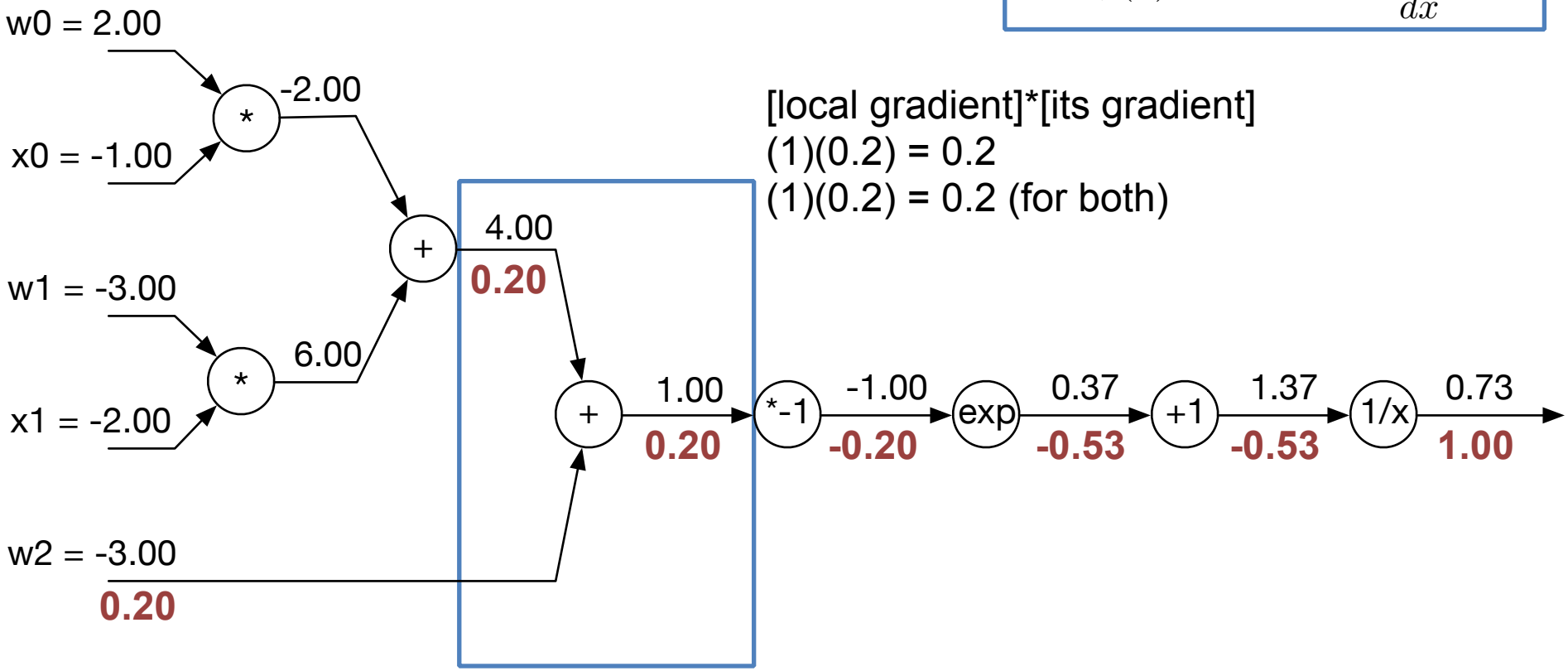
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



A second example

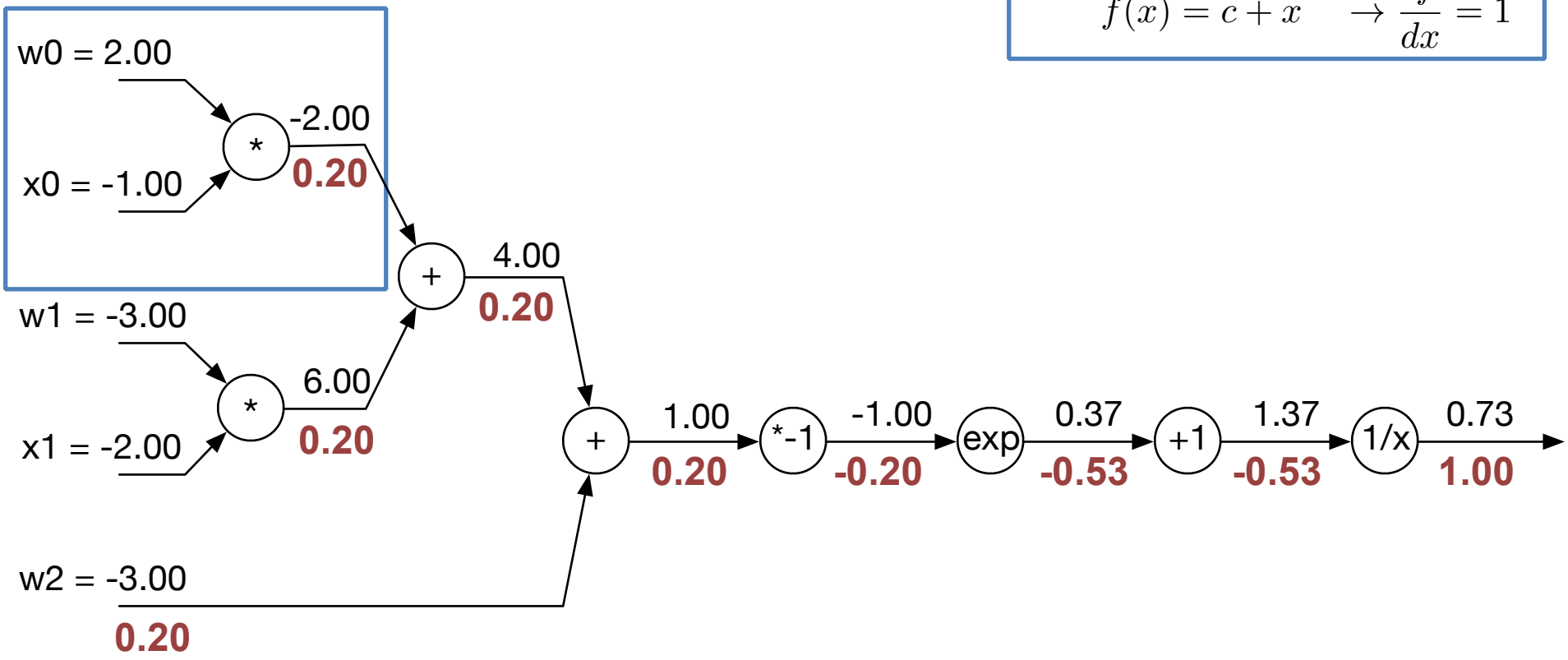
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w2)}}$$

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

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

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

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



A second example

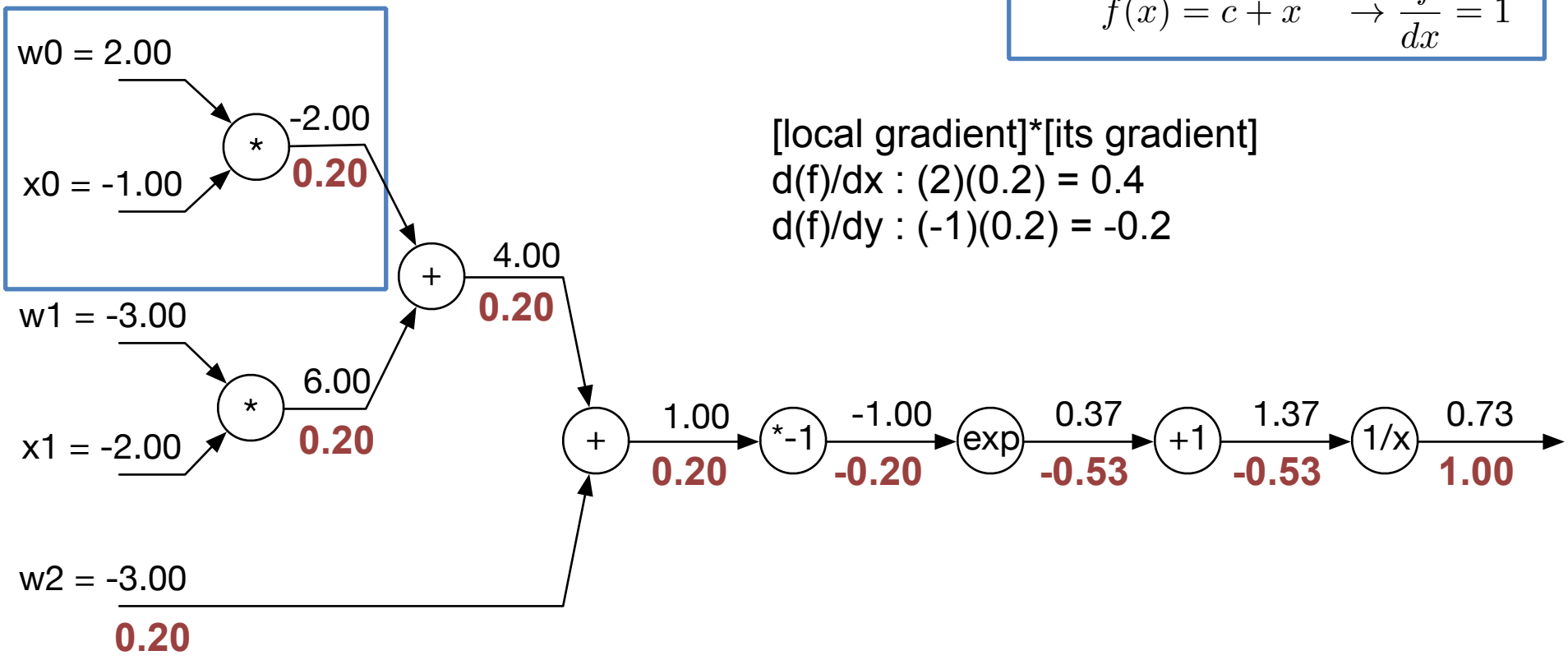
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w2)}}$$

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

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

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

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



[local gradient]*[its gradient]
d(f)/dx : (2)(0.2) = 0.4
d(f)/dy : (-1)(0.2) = -0.2

A second example

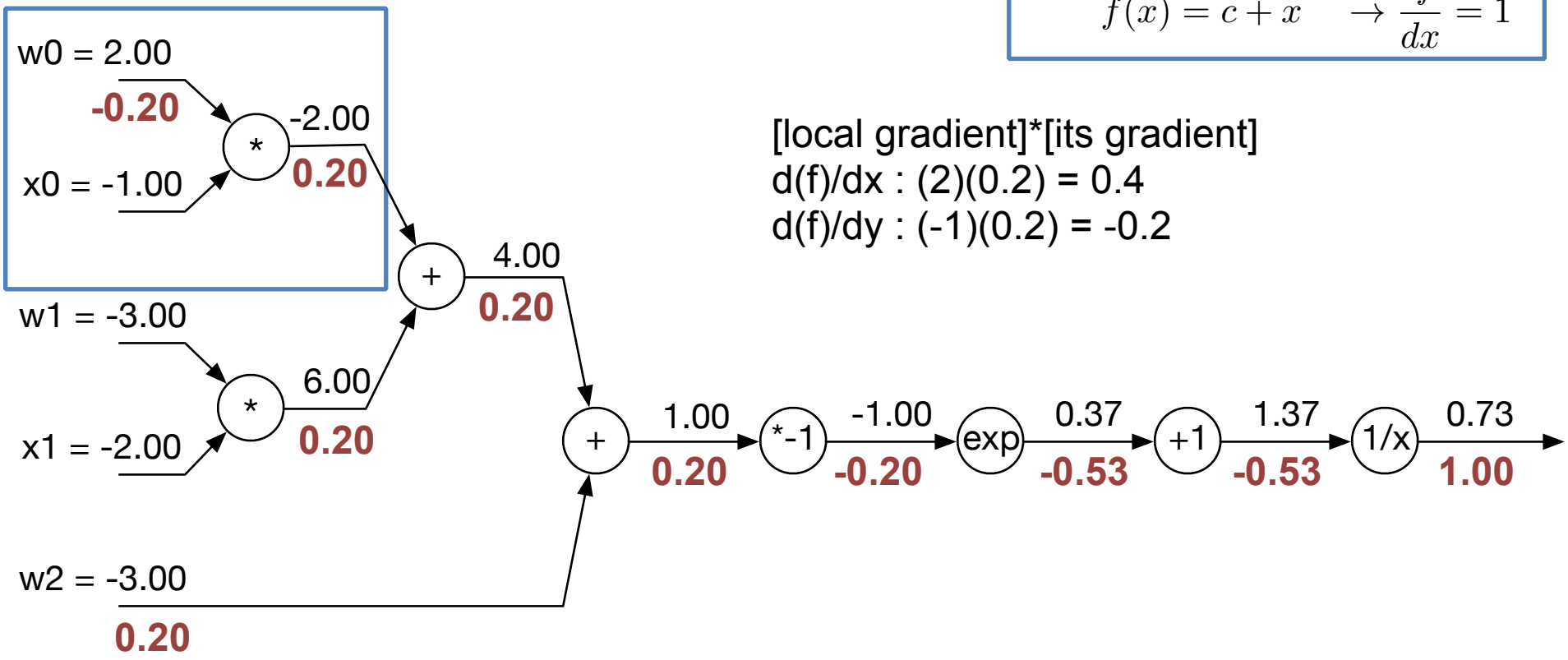
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



[local gradient]*[its gradient]
d(f)/dx : (2)(0.2) = 0.4
d(f)/dy : (-1)(0.2) = -0.2

A second example

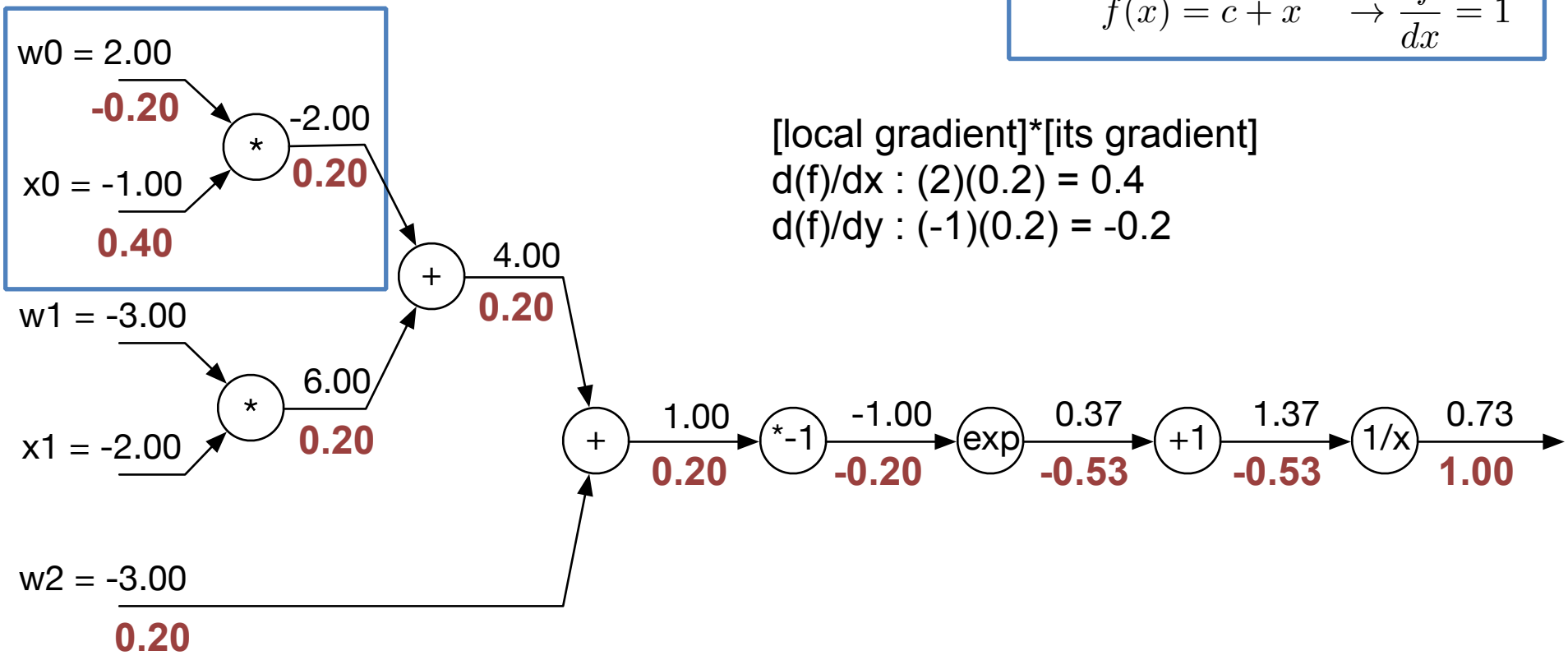
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w2)}}$$

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

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

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

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



[local gradient]*[its gradient]
d(f)/dx : (2)(0.2) = 0.4
d(f)/dy : (-1)(0.2) = -0.2

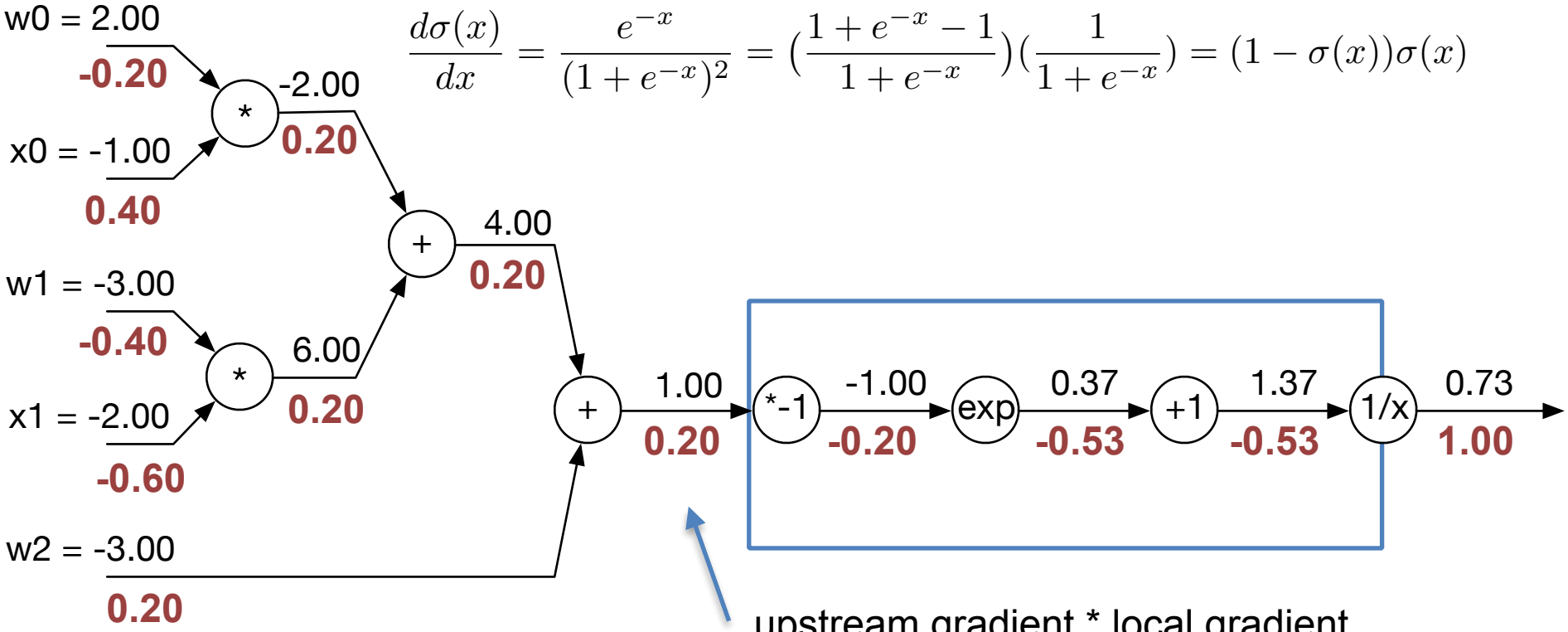
A second example

$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w2)}}$$

sigmoid function

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

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



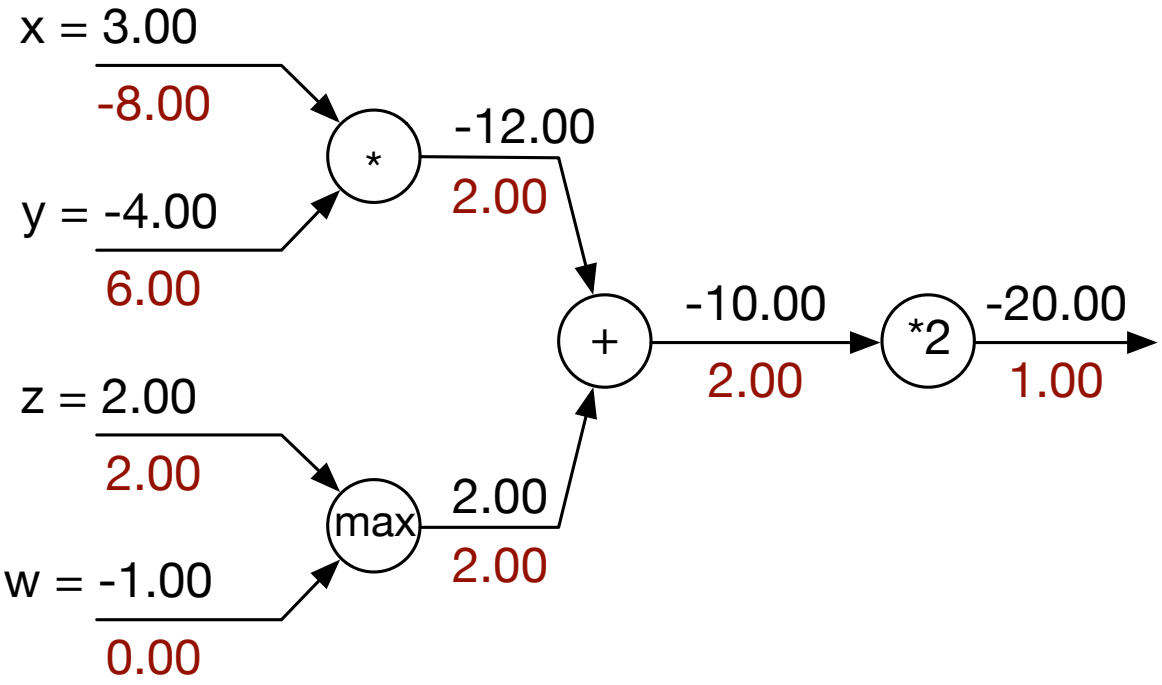
upstream gradient * local gradient
(1)*(0.73)*(1 - 0.73) = 0.2

Patterns in backward flow

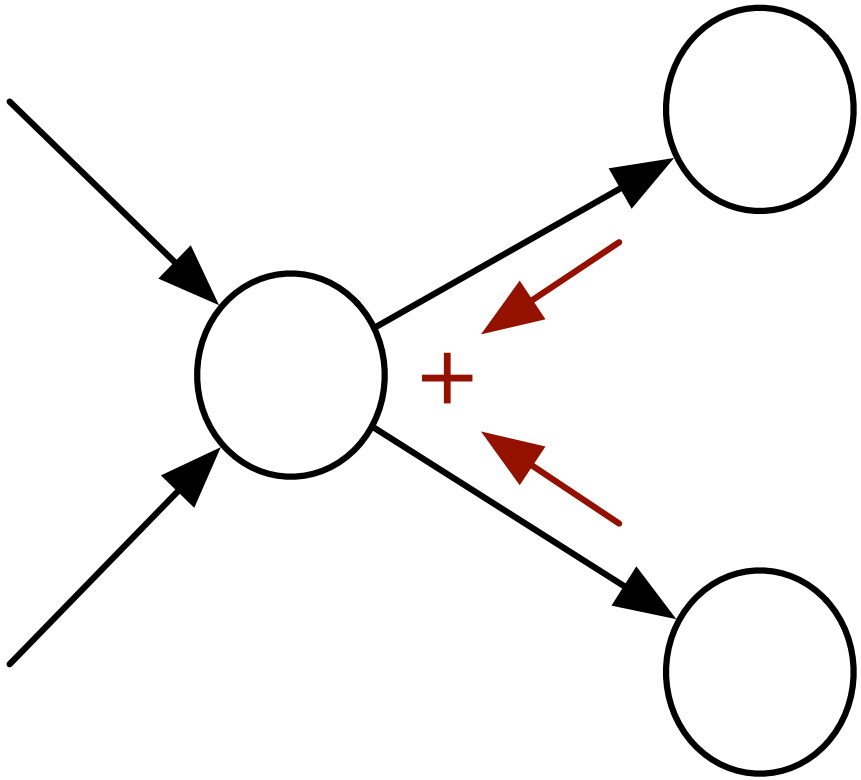
ADD gate: gradient distributor

MAX gate: gradient router

MUL gate: gradient ‘switcher’

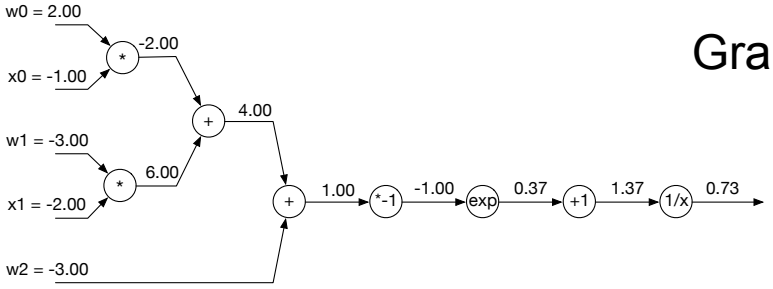


Gradients add at branches



when the one nodes sends the output to more nodes

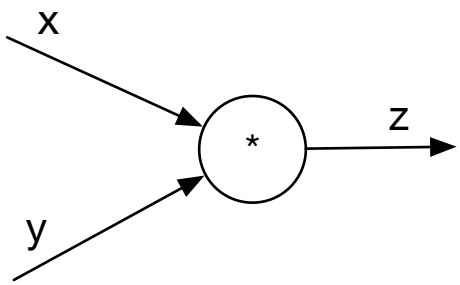
Implementation: forward/backward API



Graph (or Net) object.

```
class ComputationalGraph(object):
     #…
     def forward(inputs):
          #1. pass input to input gates
          #3. forward the computational graph
          for gate in self.graph.nodes_topologically_sorted():
               gate.forward()
          return loss #final gate in the graph
     def backward():
          for gate in reversed(self.graph.nodes_topologically_sorted()):
          gate.backward()
     return inputs_gradients
```

Implementation: forward/backward API




```
class MultiplyGate(object):
     self.x = 0
     self.y = 0
     def forward(x,y):
          z = self.x * self.y
          return z
     def backward(dLdz):
          dLdx = self.y * dLdz;
          dLdy = self.x * dLdz;
     return dLdx, dLdy
```

$$\frac{\partial L}{dz}$$

Note: this example shows a gate that handles scalar. In most implementations, vectorised inputs are used to take advantage of parallel matrix multiplications

Example: Torch, MulConstant

```
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
```

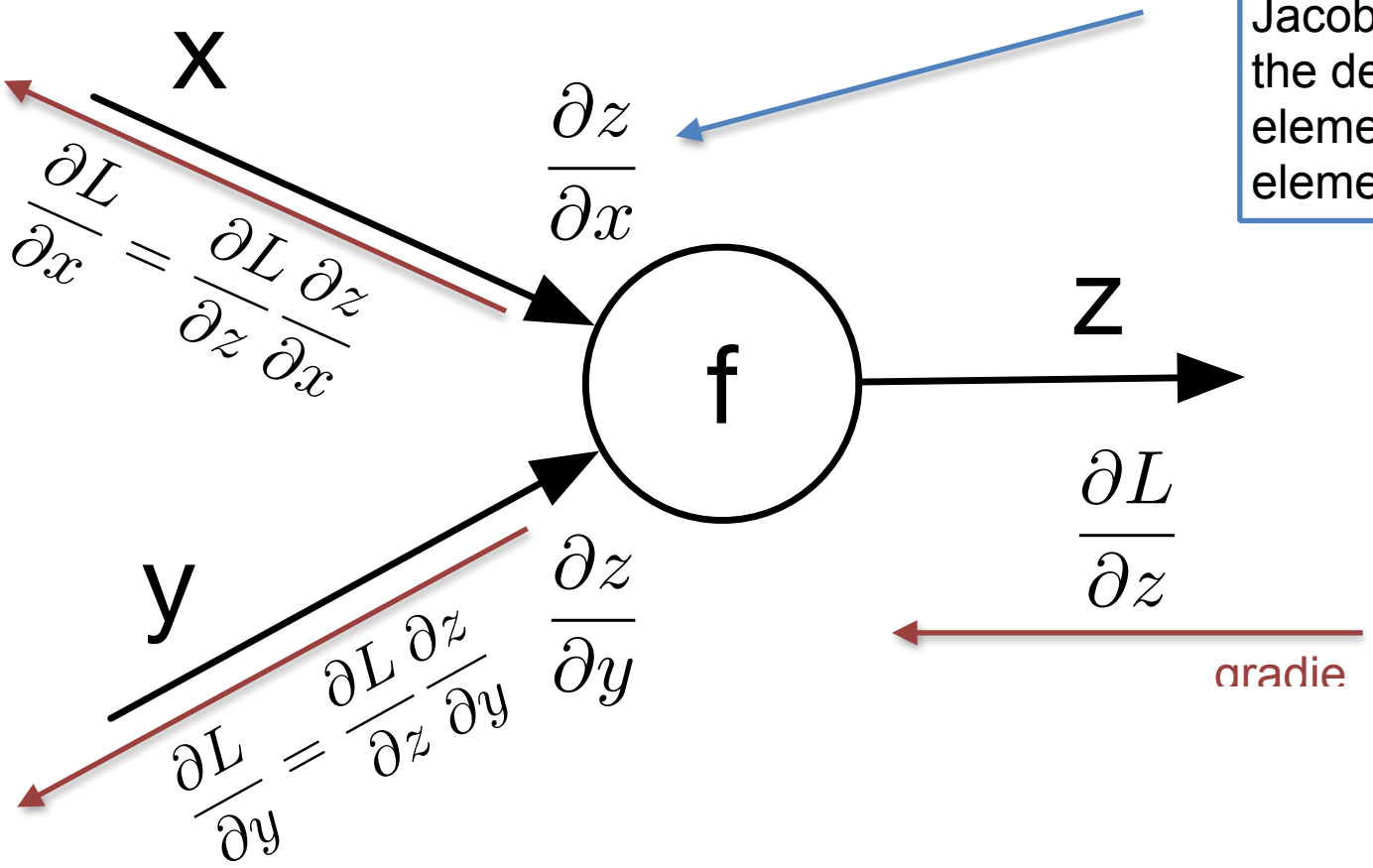
$f(x) = aX$

initialization

forward pass

backward pass

Working with vectors

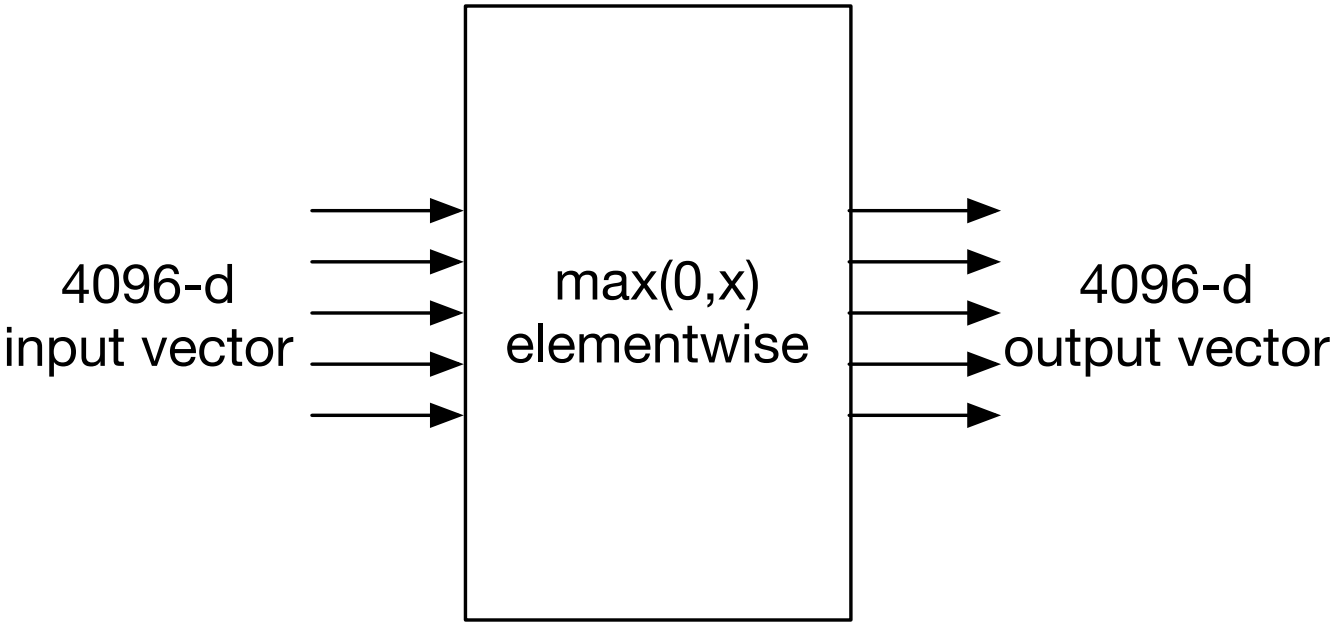


This is now the Jacobian matrix, i.e. the derivative of each element of z w.r.t. each element of x

Vectorised operations

What is the size of the Jacobian matrix?

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



Jacobian matrix: 4096x4096, but we this matrix has a special structure: it's a diagonal matrix with only 1 or 0 on the diagonal, and 0 elsewhere.

III Putting it all together

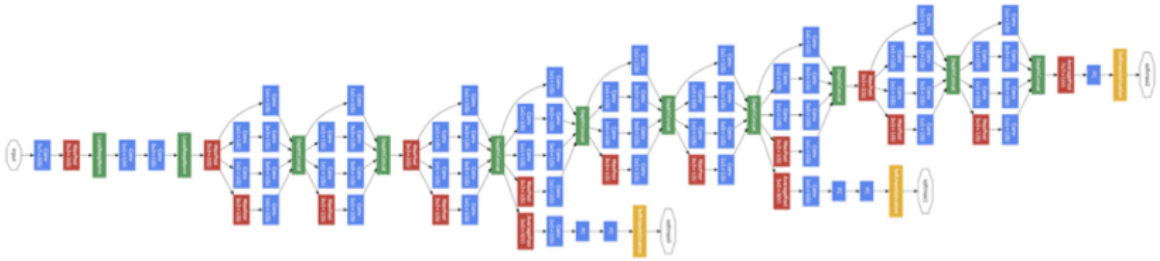
A multi-layer neural network is like a structure built from basic blocks.

Each block is capable of performing a **forward pass** (from input to output) and a **backward pass** (backpropagating the gradient).

Each block is also optimised to perform vectorised operations.

A computational graph keeps track of the order of computation both in the forward and in the backward pass.

In this way we can compute a complex sequence of operation in very large networks



inception network

Convolution
Pooling
Softmax

Example of a code for a 4-layer convolutional ANN

```python
from .network_utils import *

class NatureConvBody(nn.Module):
    def __init__(self, in_channels=1):
        super(NatureConvBody, self).__init__()
        self.feature_dim = 128
        self.conv1 = layer_init(nn.Conv2d(in_channels, 12, kernel_size=2,
        stride=1))
        self.conv2 = layer_init(nn.Conv2d(12, 12, kernel_size=4, stride=2))
        self.conv3 = layer_init(nn.Conv2d(12, 16, kernel_size=3, stride=1))
        self.fc4 = layer_init(nn.Linear(64, self.feature_dim))

    def forward(self, x):
        y = F.relu(self.conv1(x))
        y = F.relu(self.conv2(y))
        y = F.relu(self.conv3(y))
        y = y.view(y.size(0), -1)
        y = F.relu(self.fc4(y))
        return y
```

Note that
- the layers are defined in the init block
- the computational graph is described in the forward pass
- the backward operation is implicit (i.e. the library knows)

```python
loss = self.criterion(q, q_next)
self.optimizer.zero_grad()
loss.backward()
nn.utils.clip_grad_norm_(self.network.parameters(),
self.config.gradient_clip)
self.optimizer.step()
```

the loss is computed

gradients are reset

new gradients are computed

one step in the optimisation is performed

Summary

This week (W2) lectures:

- introduced the coursework

- overviewed some very general principles of biological neural networks

- introduced the concept of artificial neuron and perceptron

- discussed the problem of classification and introduced a loss function

- introduced the concept of backpropagating gradients, a concept used to train most neural networks and called backpropagation