

ELEC 576: Neural Networks & Backpropagation Lecture 3

Ankit B. Patel

Baylor College of Medicine (Neuroscience Dept.)

Rice University (ECE Dept.)

09-06-2016

Administrivia

- We've increased the limit in the class (again)
- Assignment #0 will be available on website this evening (not for credit but very *very* important!)
- Register for the class on Gradescope (detailed instructions in Asst #0)
- Let's take a picture to let the Registrar know we need more space!

Latest News

TF-Slim: A High Level Library to Define Complex Models in Tensorflow

<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim>



Google Research Blog

The latest news from Research at Google

TF-Slim: A high level library to define complex models in TensorFlow

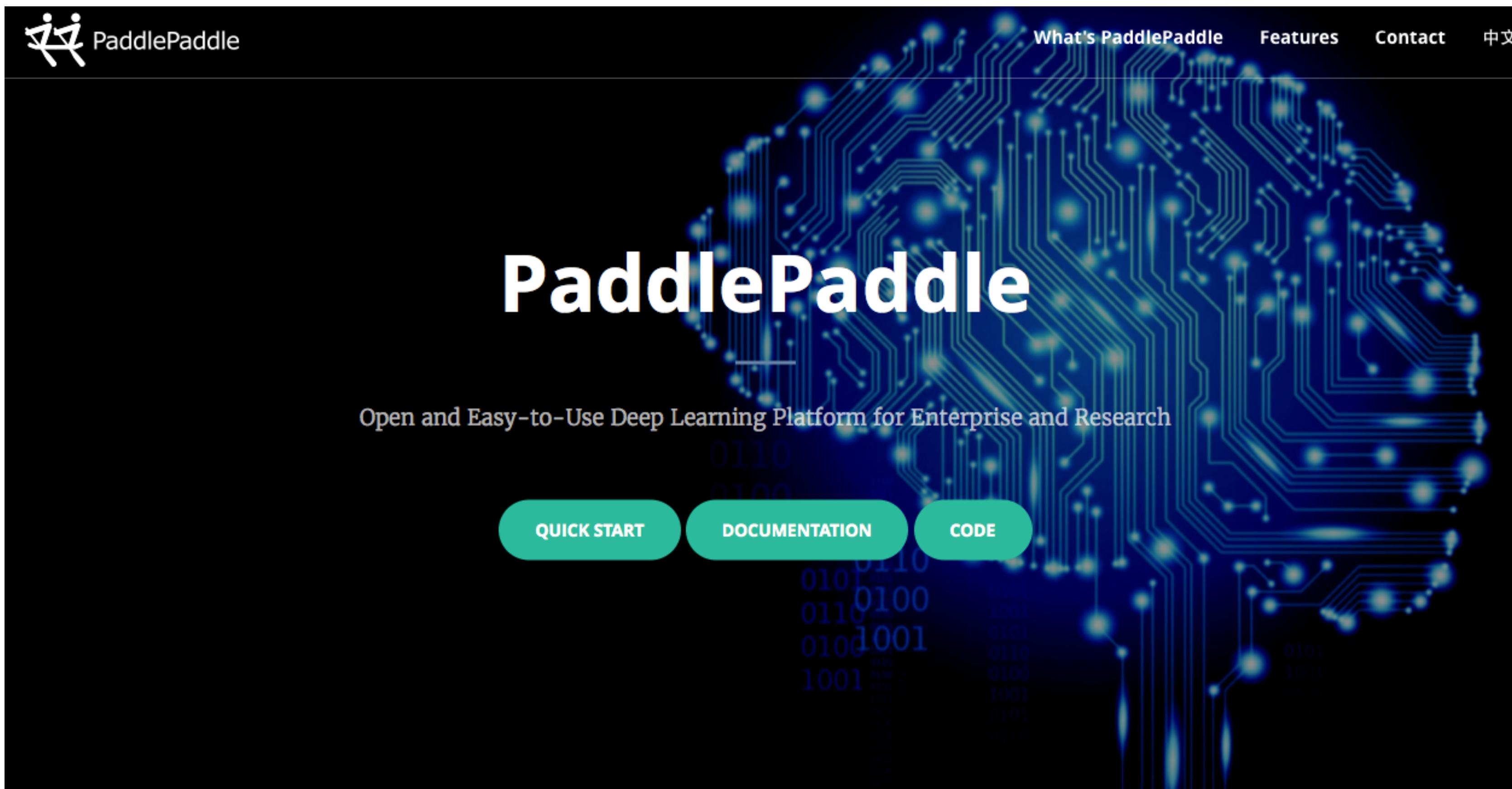
Tuesday, August 30, 2016

Posted by Nathan Silberman and Sergio Guadarrama, Google Research

Earlier this year, we released a TensorFlow implementation of a state-of-the-art image classification model known as [Inception-V3](#). This code allowed users to train the model on the [ImageNet classification dataset](#) via synchronized gradient descent, using either a single local machine or a cluster of machines. The Inception-V3 model was built on an experimental [TensorFlow](#) library called [TF-Slim](#), a lightweight package for defining, training and evaluating models in TensorFlow. The TF-Slim library provides common abstractions which enable users to define models quickly and concisely, while keeping the model architecture transparent and its hyperparameters explicit.

Baidu Release Its Open-source Platform

<https://github.com/baidu/paddle>



Self-Driving Cars: Tesla Autopilot

- NYTimes Video
- <https://www.youtube.com/watch?v=5TjbqVartjM>

Tesla Autopilot Accident

...due to a misclassification error



Why?

- **Mis-Classification:** Visual perception system misclassified tractor trailer as billboard/overhead road sign
- **Depth Ambiguity:** radar system also ignored truck because otherwise it leads to too many false positive braking events (overhead road signs)
- **Video Tracking/Dynamical Modeling:** system did not translate knowledge from earlier in time (that there was indeed a tractor trailer moving across field of view) into current understanding

Potential Solutions (Project Ideas)

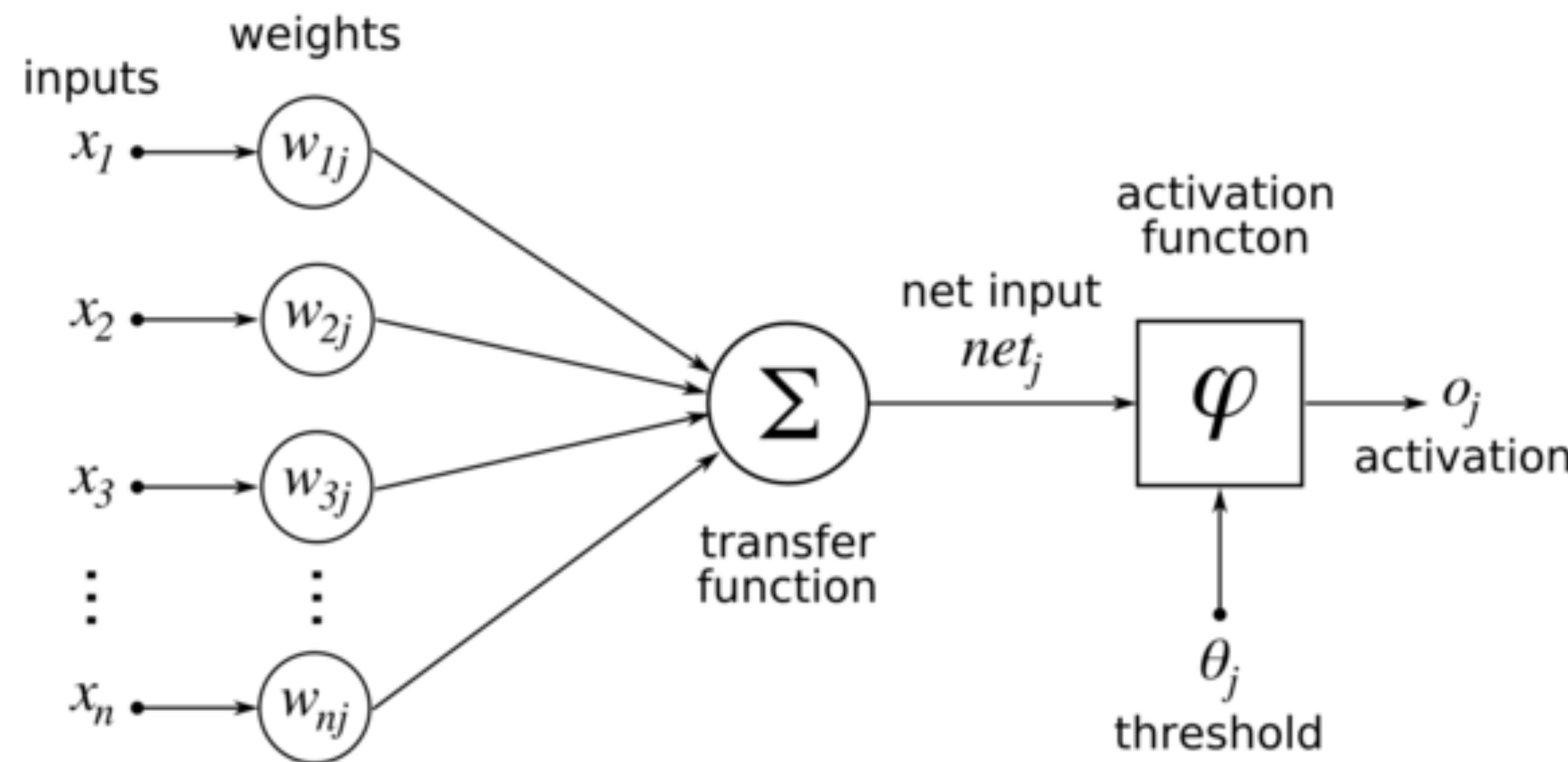
- **Lesson:** Good performance on tasks is not sufficient. It is absolutely critical that we understand why our NNs make the decisions they do — we need human-interpretable explanations for their decisions.
- **Project Ideas:**
 - How to probe trained NNs to see what factors led it to its decision?
Applications in Self-Driving Cars, Medical Imaging, etc.
 - Dynamical Modeling/Video Understanding with Convnets
 - Depth Perception Jointly with Image Classification

Outline

- Neural Networks
 - Definition of NN and terminology
- Review of (Old) Theoretical Results about NNs
 - Intuition for why compositions of nonlinear functions are more expressive
 - Expressive power theorems [McC-Pitts, Rosenblatt, Cybenko]
- Backpropagation algorithm (Gradient Descent + Chain Rule)
 - History of backdrop summary
 - Gradient descent (Review).
 - Chain Rule (Review).
 - Backprop
- Intro to Convnets
 - Convolutional Layer, ReLu, Max-Pooling

Neural Networks

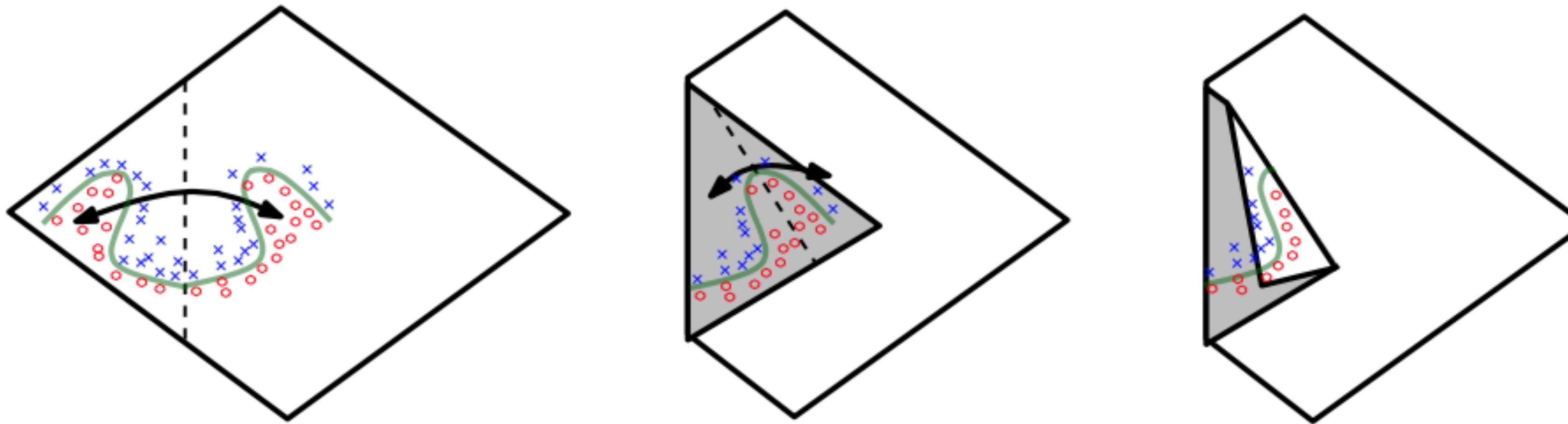
Neural Network



Expressive Power Theorems

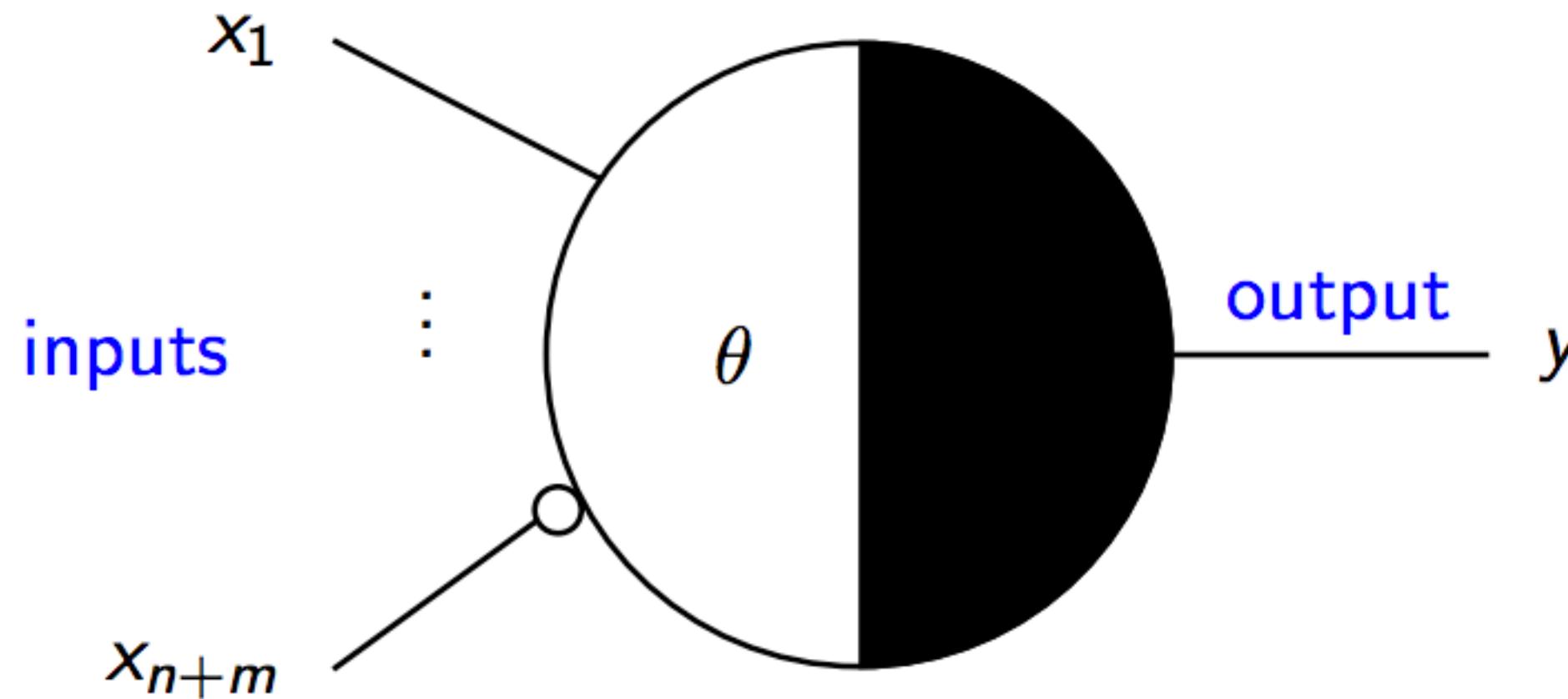
Compositions of Nonlinear Functions are more expressive

Deeper nets with rectifier/maxout units are exponentially more expressive than shallow ones (1 hidden layer) because they can split the input space in many more (not-independent) linear regions, with constraints, e.g., with abs units, each unit creates mirror responses, folding the input space:



[Yoshua Bengio]

McCulloch-Pitts Neurons



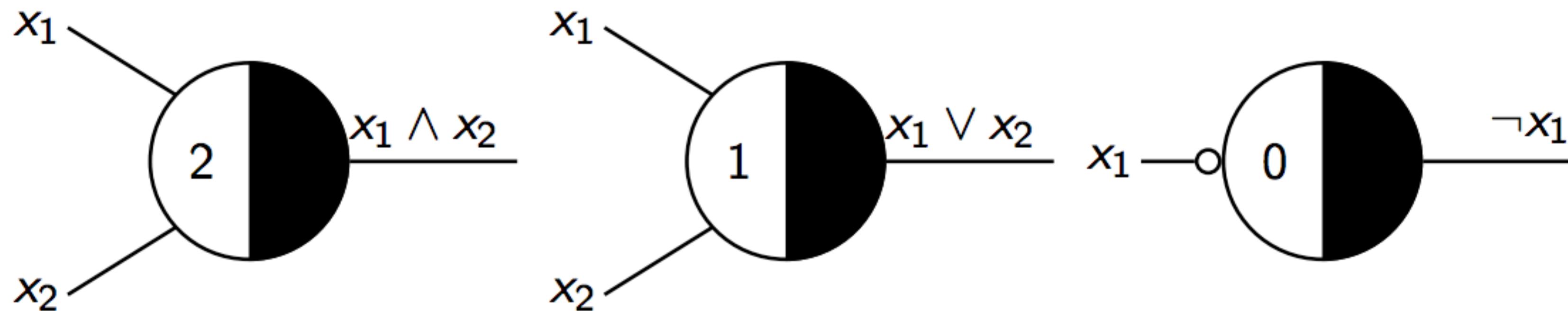
Stefan Droste

- all signals binary ($\in \{0, 1\}$)
- threshold $\theta \in \mathbb{N}_0$
- n excitatory inputs x_1, \dots, x_n
- m inhibitory inputs x_{n+1}, \dots, x_{n+m}
- one output y (with unrestricted fan-out)

$$y(x_1, \dots, x_{n+m}, \theta) = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i \geq \theta \text{ and } \sum_{i=n+1}^m x_i = 0 \\ 0 & \text{if } \sum_{i=1}^n x_i < \theta \text{ or } \sum_{i=n+1}^m x_i > 0 \end{cases}$$

Expressive Power of McCulloch-Pitts Nets

Feed-forward McCulloch-Pitts nets can compute any Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$.

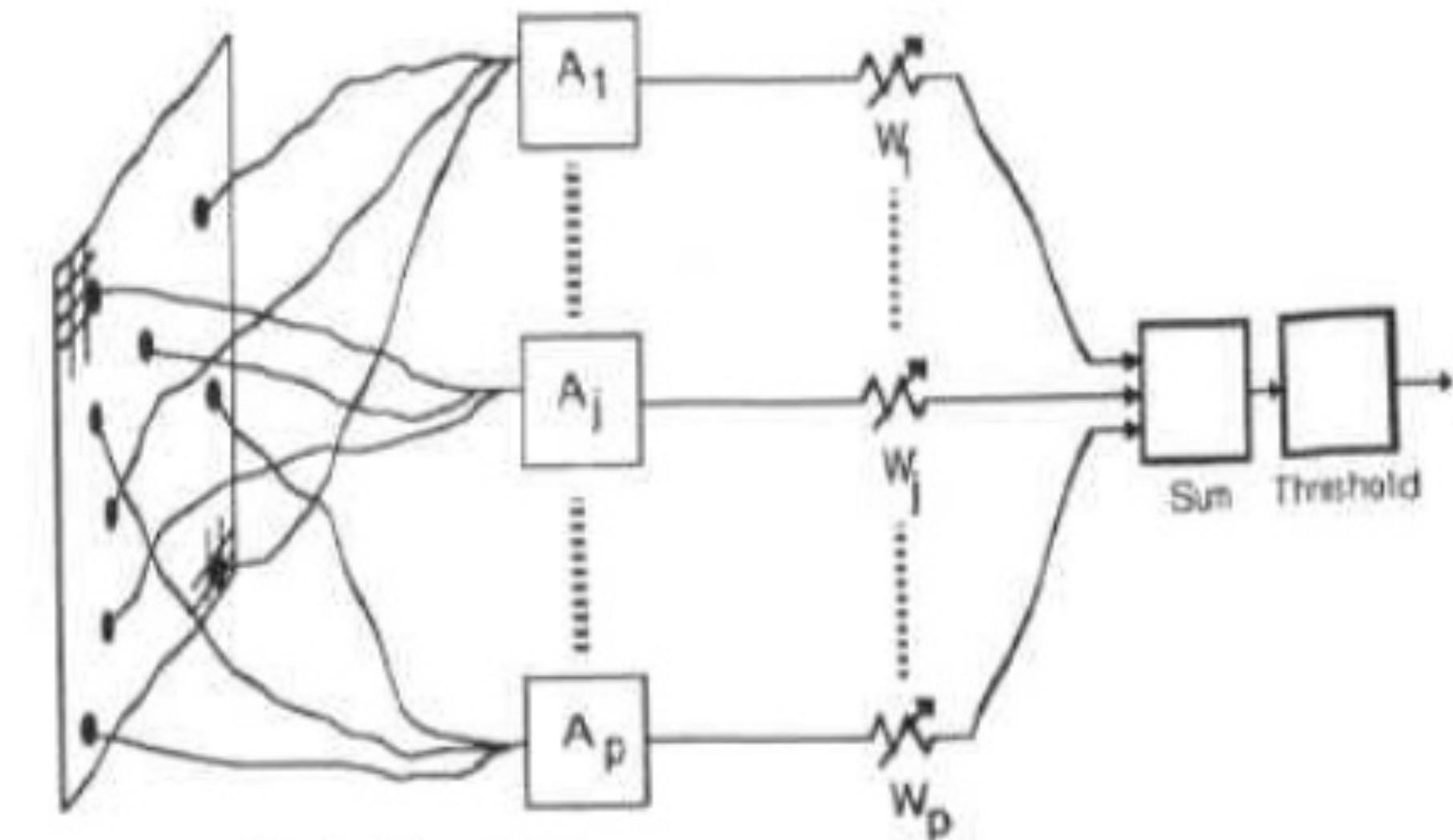


Stefan Droste

Recursive McCulloch-Pitts nets can simulate any deterministic finite automaton (DFA).

The Perceptron (Rosenblatt)

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

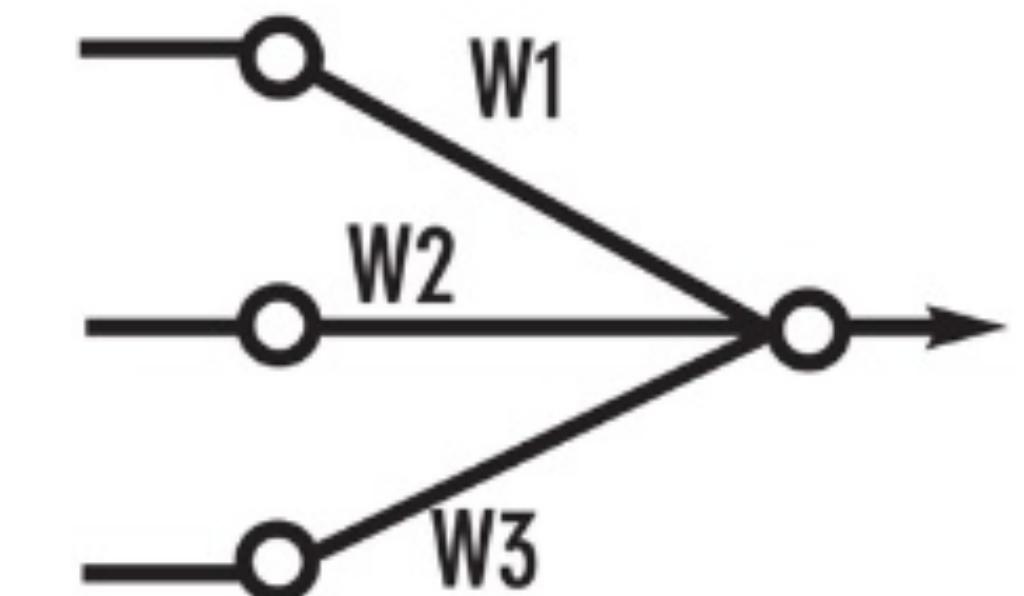


Original Perceptron

(From *Perceptrons* by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.



Frank Rosenblatt
(1928-1971)



Simplified model:

Limitations of Perceptron

- Rosenblatt was overly enthusiastic about the perceptron and made the ill-timed proclamation that:
- "Given an elementary α -perceptron, a stimulus world W , and any classification $C(W)$ for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to $C(W)$ in finite time..." [4]
- In 1969, Marvin Minsky and Seymour Papert showed that the perceptron could only solve linearly separable functions. Of particular interest was the fact that the perceptron still could not solve the XOR and NXOR functions.
- Problem outlined by Minsky and Papert can be solved by deep NNs. However, many of the artificial neural networks in use today still stem from the early advances of the McCulloch-Pitts neuron and the Rosenblatt perceptron.

Universal Approximation Theorem

[Cybenko 1989, Hornik 1991]

Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote the m -dimensional unit hypercube $[0, 1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exists an integer N , real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, where $i = 1, \dots, N$, such that we may define:

$$F(x) = \sum_{i=1}^N v_i \varphi(w_i^T x + b_i)$$

seem as slope of linear approximation

determine which piece interval

as an approximate realization of the function f where f is independent of φ ; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

This still holds when replacing I_m with any compact subset of \mathbb{R}^m .

- https://en.wikipedia.org/wiki/Universal_approximation_theorem

- Problems:
1. N could be super high.
 2. Not appropriate for high dimension functions
 3. A lot of parameters, not learnable

Universal Approximation Theorem

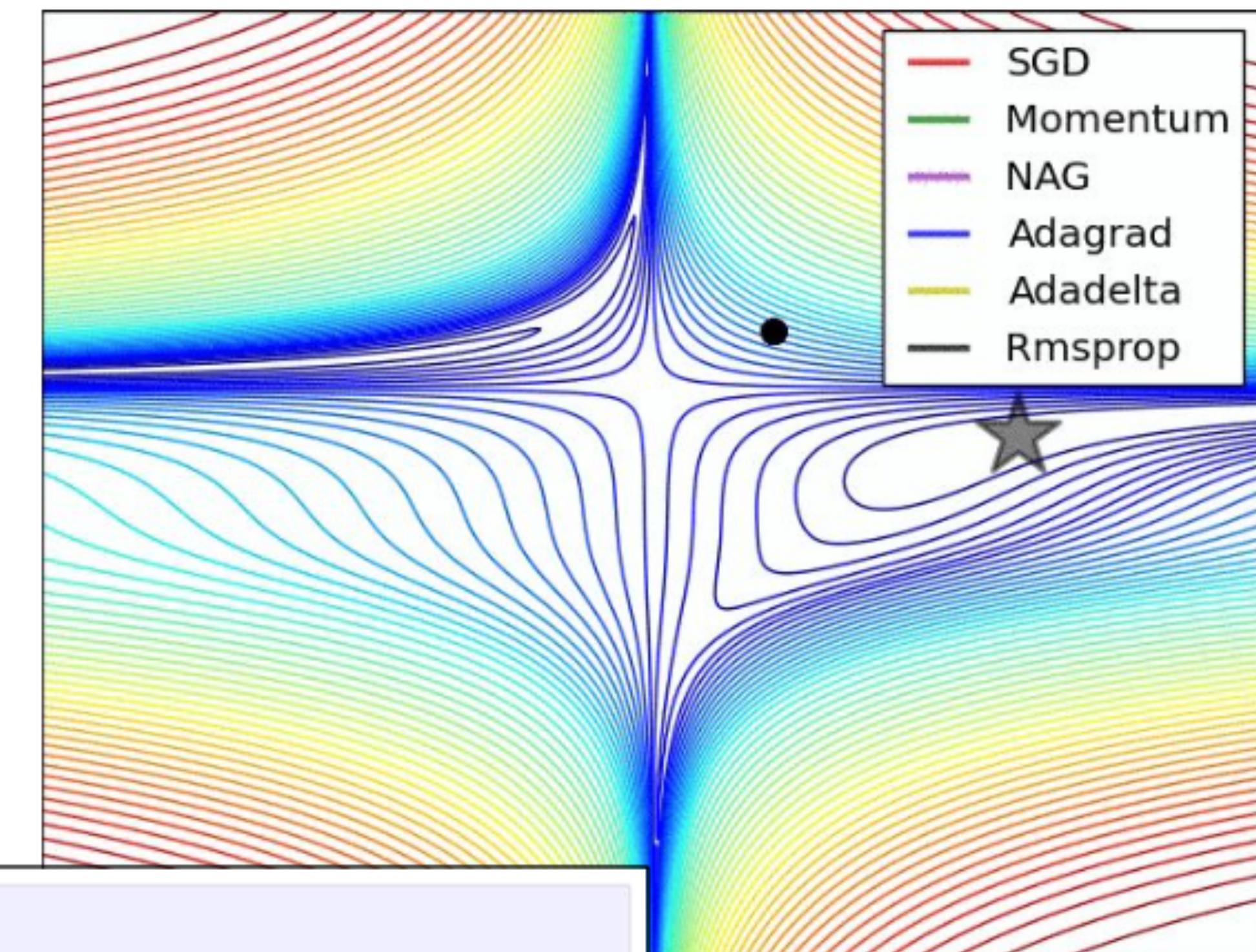
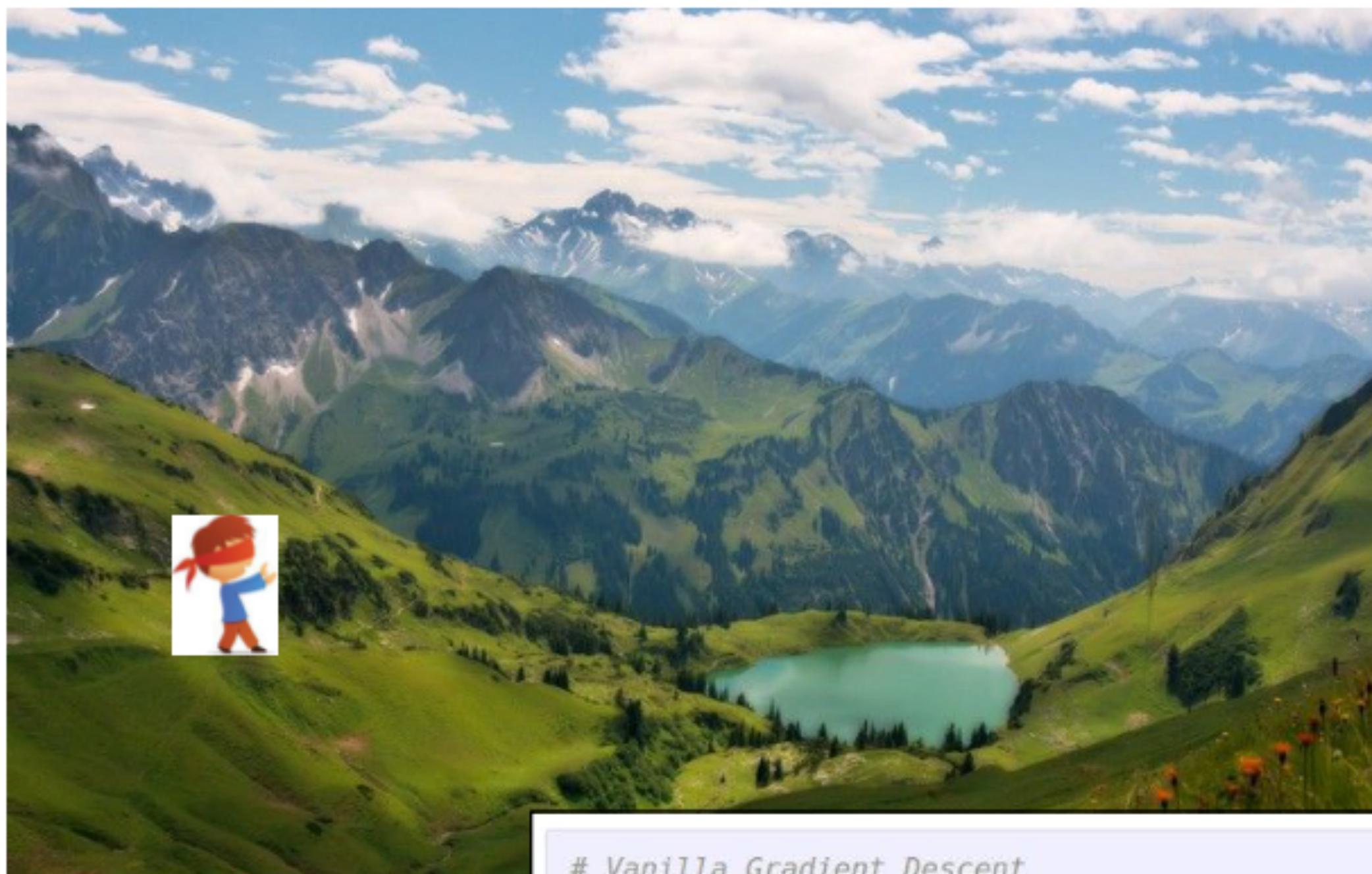
- https://en.wikipedia.org/wiki/Universal_approximation_theorem
- Shallow neural networks can *represent* a wide variety of interesting functions when given appropriate parameters; however, it does not touch upon the algorithmic **learnability** of those parameters.
- Proved by **George Cybenko** in 1989 for **sigmoid** activation functions.^[2]
- Kurt Hornik showed in 1991^[3] that it is not the specific choice of the activation function, but rather the multilayer feedforward architecture itself which gives neural networks the potential of being universal approximators.

Why we need nonlinear to proximate? Intuitive: for example, x^2 . Iterative/ go deep could produce nonlinear with only limit parameters.

Backpropagation

Gradient Descent

Optimization



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

(image credits
to Alec Radford)

[Fei-Fei Li, Andrej Karpathy, Justin Johnson]

How to Learn NNs? History of the Backpropagation Algorithm (1960-86)

- Introduced by Henrey J. Kelley (1960) and Arthur Bryson (1961) in control theory, using Dynamic Programming
- Simpler derivation using **Chain Rule** by Stephen Dreyfus (1962)
- General method for Automatic Differentiation by Seppo Linnainamaa (1970)
- Using backdrop for parameters of controllers minimizing error by Stuart Dreyfus (1973)
- Backprop brought into NN world by Paul Werbos (1974)
- Used it to learn representations in hidden layers of NNs by Rumelhart, Hinton & Williams (1986)

Gradient Descent

- Optimization problem

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w})$$

- Insight: At the point \mathbf{a} , $f(\mathbf{w})$ decreases fastest in the direction of the negative gradient of f (assuming f is defined and differentiable at \mathbf{a})

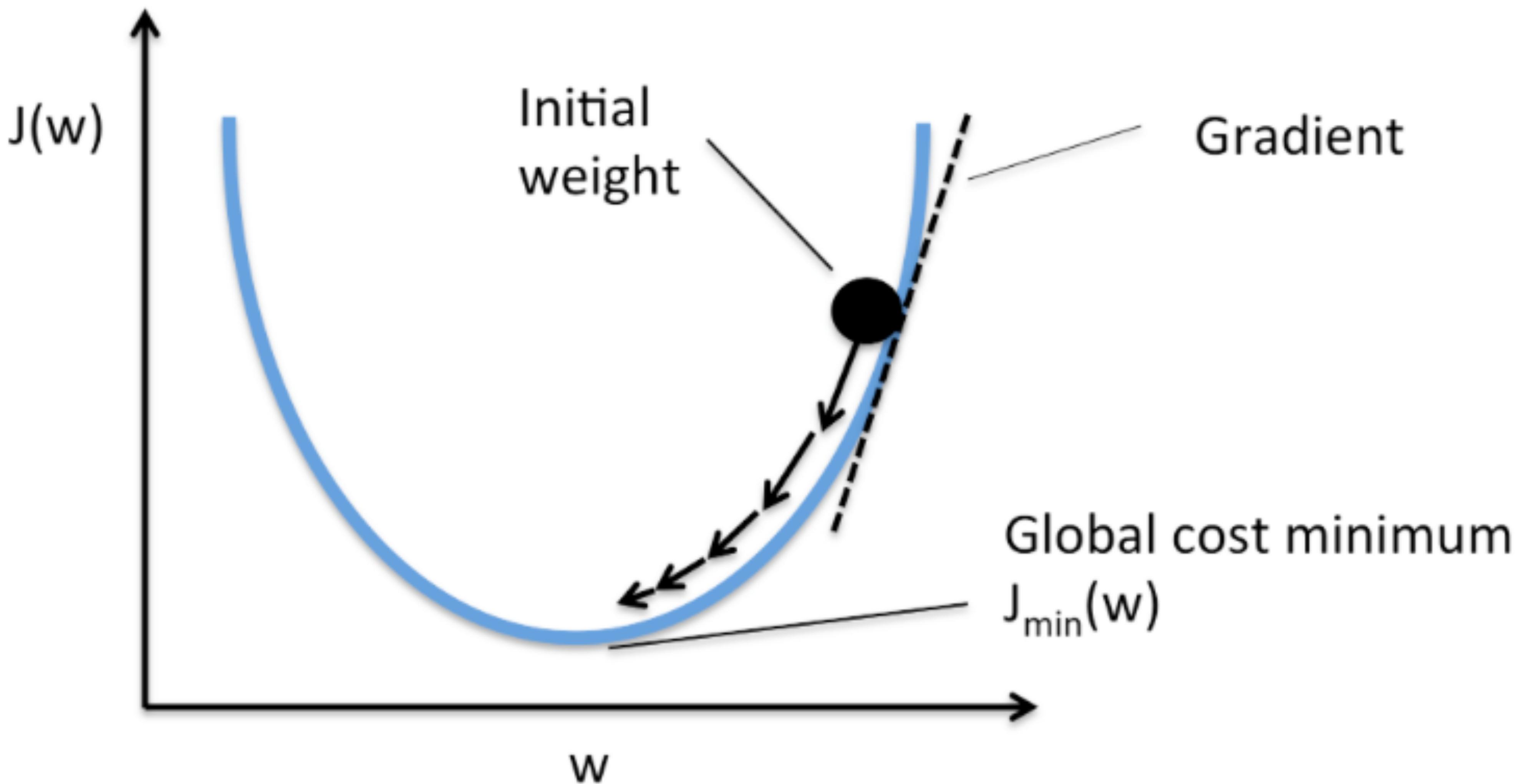
$$-\nabla f(\mathbf{a})$$

- Step in the direction of the negative gradient

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \mu^t \nabla f(\mathbf{w}^t)$$

where μ^t is the step size at step t

Gradient Descent



Interesting Question:

What kind of problems might you run
into with Gradient Descent? (2 min)

for non-differentiate function: try kernel smoothing before calculating gradient.

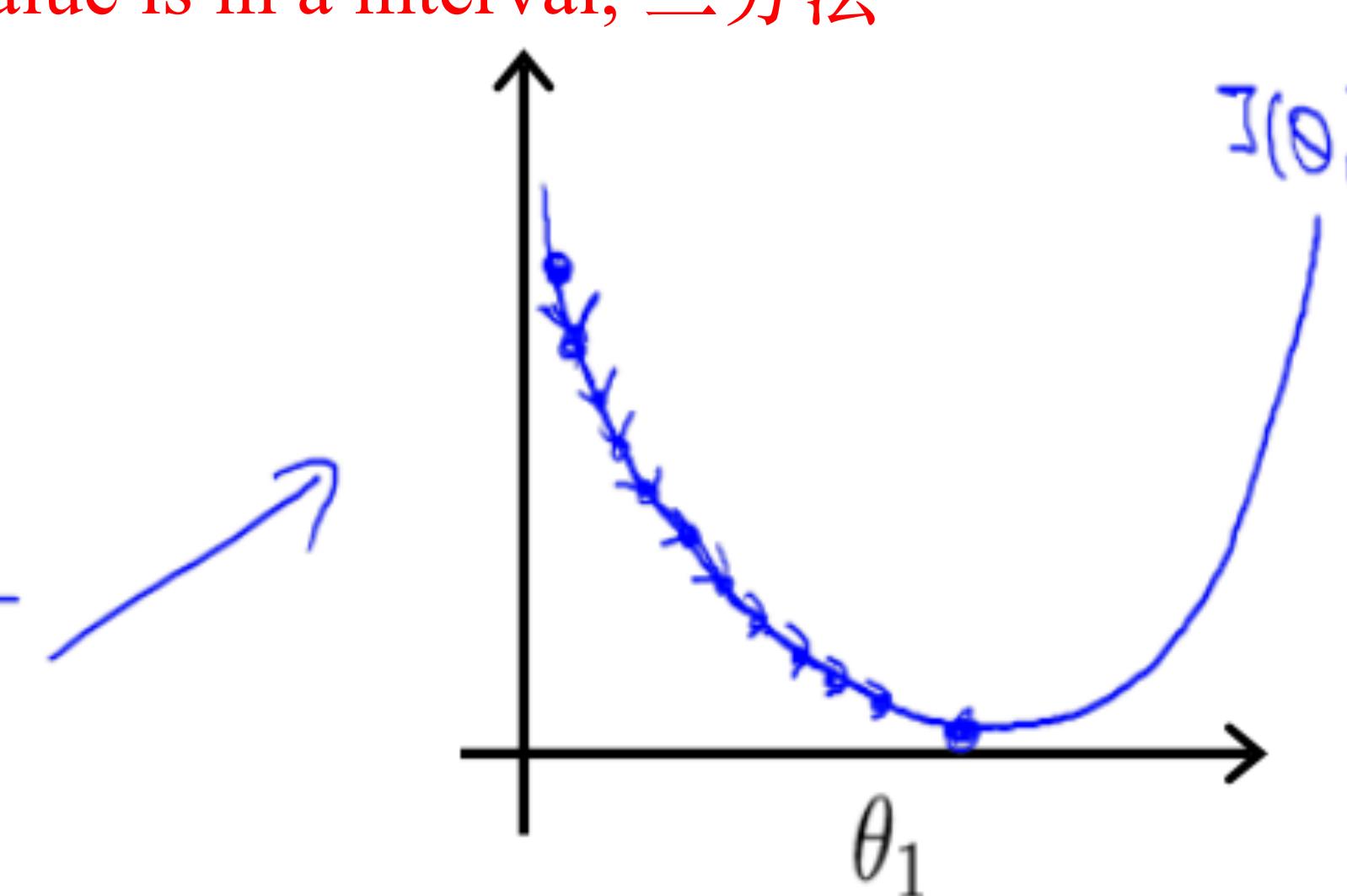
- 1.
- 2.saddle point

Learning Rate Needs to Be Carefully Chosen

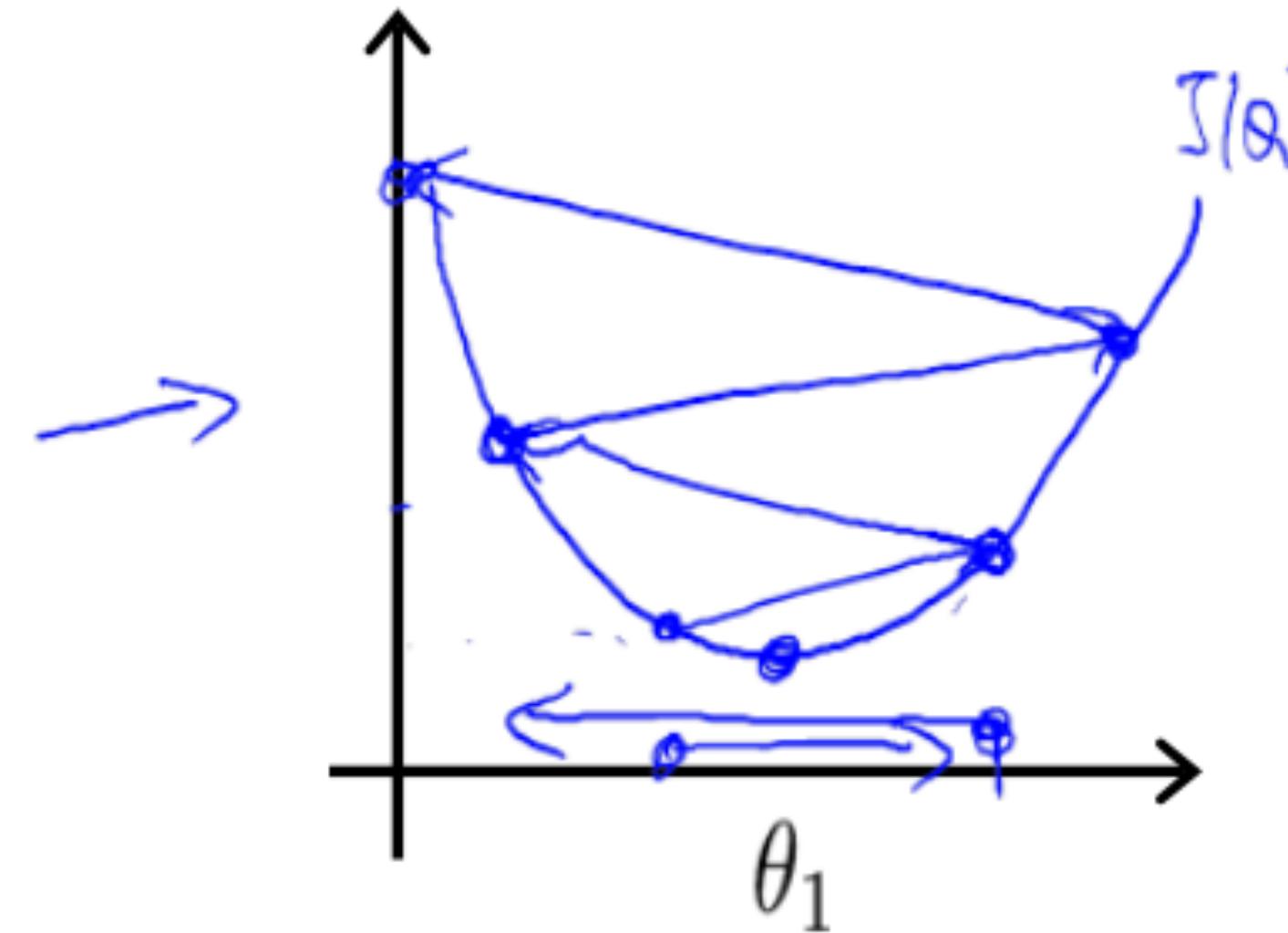
Binary Search: to adjust step size, the best value is in a interval, 二分法

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

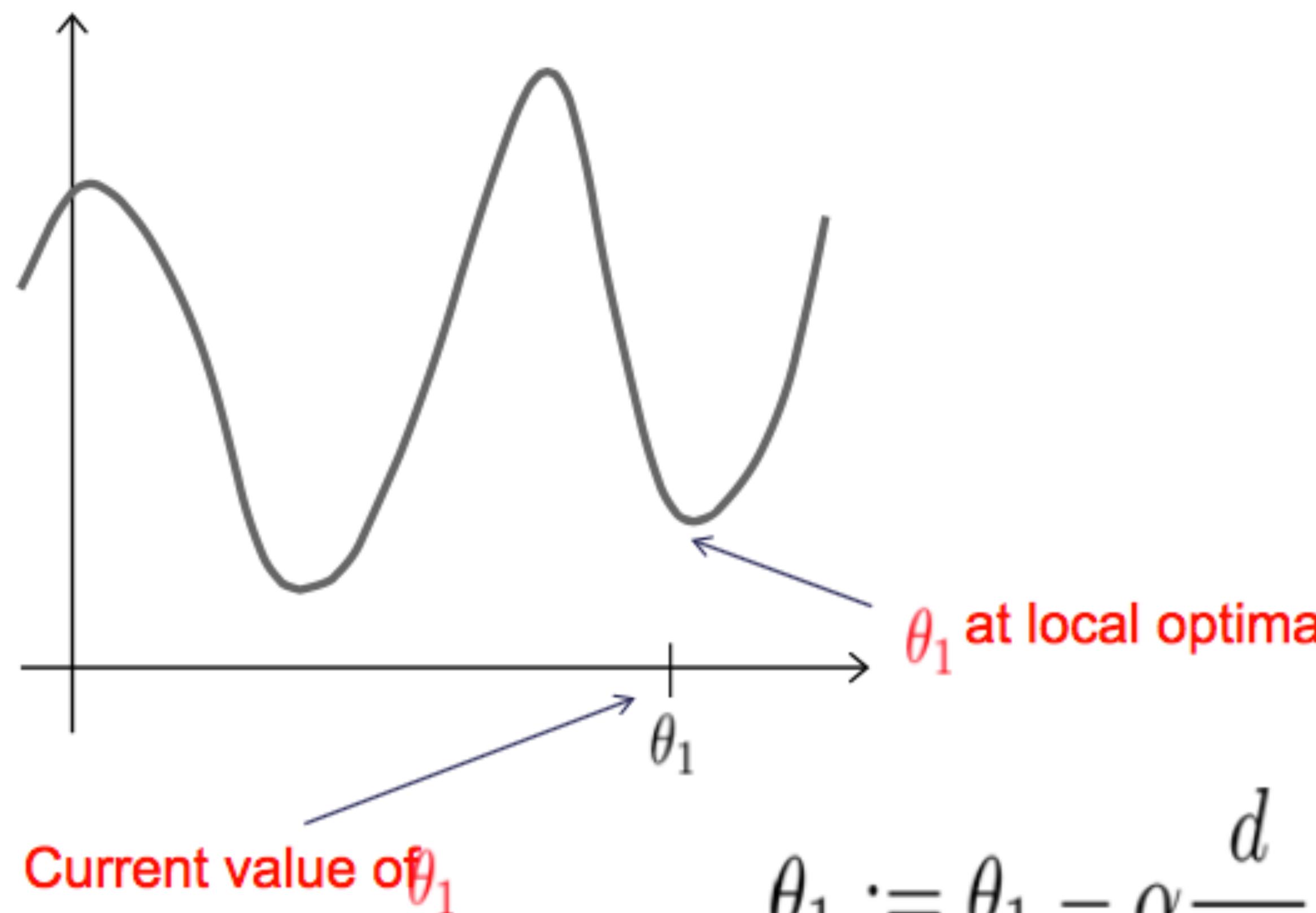
If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Global Optima is not Guaranteed



$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Chain Rule

If f and g are both differentiable and F is the composite function defined by $F(x)=f(g(x))$, then F is differentiable and F' is given by the product

$$F'(x) = f'(g(x)) \cdot g'(x)$$

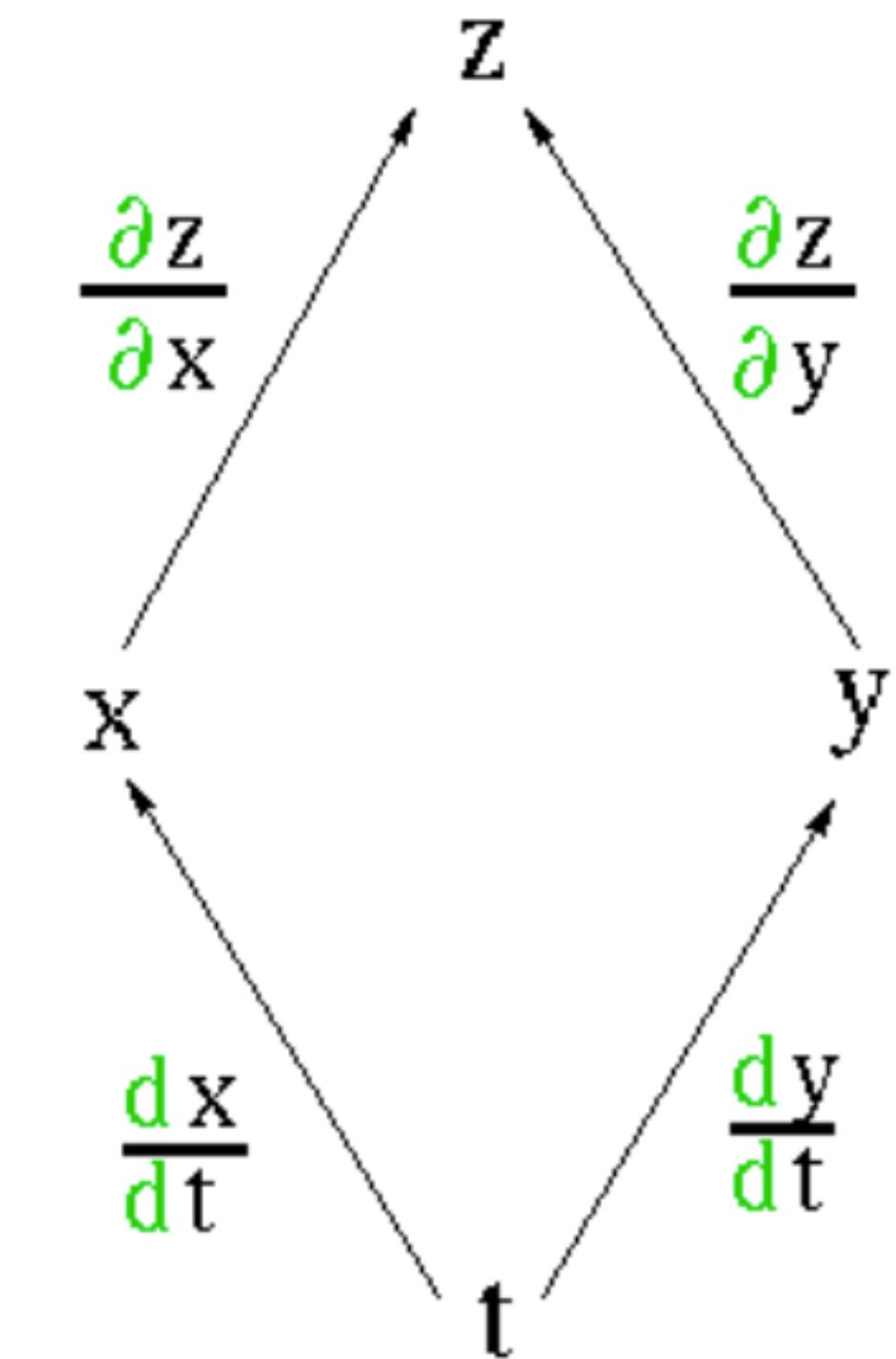
In Leibniz notation, if $y=f(u)$ and $u=g(x)$ are both differentiable functions, then

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$

Chain Rule

Let $x = x(t)$ and $y = y(t)$ be differentiable at t and suppose that $z = f(x, y)$ is differentiable at the point $(x(t), y(t))$. Then $z = f(x(t), y(t))$ is differentiable at t and

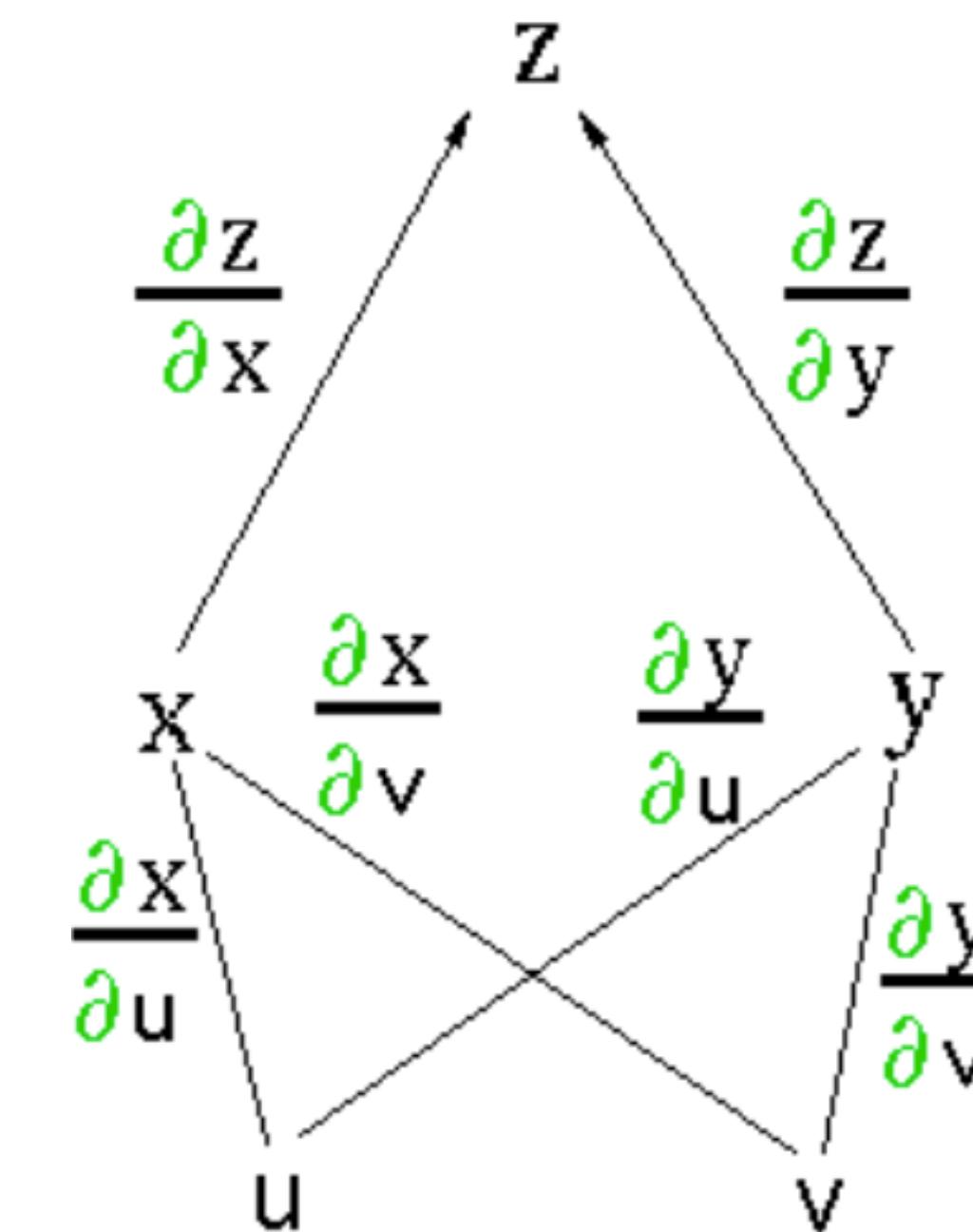
$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}.$$



Chain Rule

Let $x = x(u, v)$ and $y = y(u, v)$ have first-order partial derivatives at the point (u, v) and suppose that $z = f(x, y)$ is differentiable at the point $(x(u, v), y(u, v))$. Then $f(x(u, v), y(u, v))$ has first-order partial derivatives at (u, v) given by

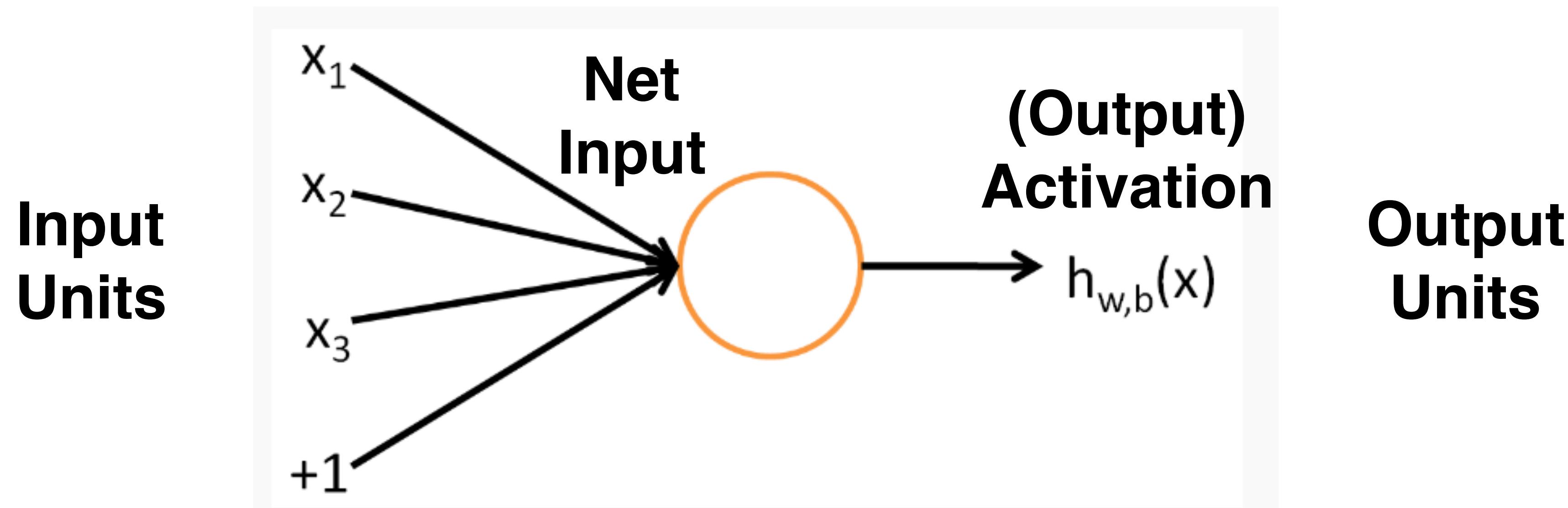
$$\begin{aligned}\frac{\partial z}{\partial u} &= \frac{\partial z}{\partial x} \frac{\partial x}{\partial u} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial v} &= \frac{\partial z}{\partial x} \frac{\partial x}{\partial v} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial v}.\end{aligned}$$



Fun Exercise:
Do Chain Rule on nested function (5
min)

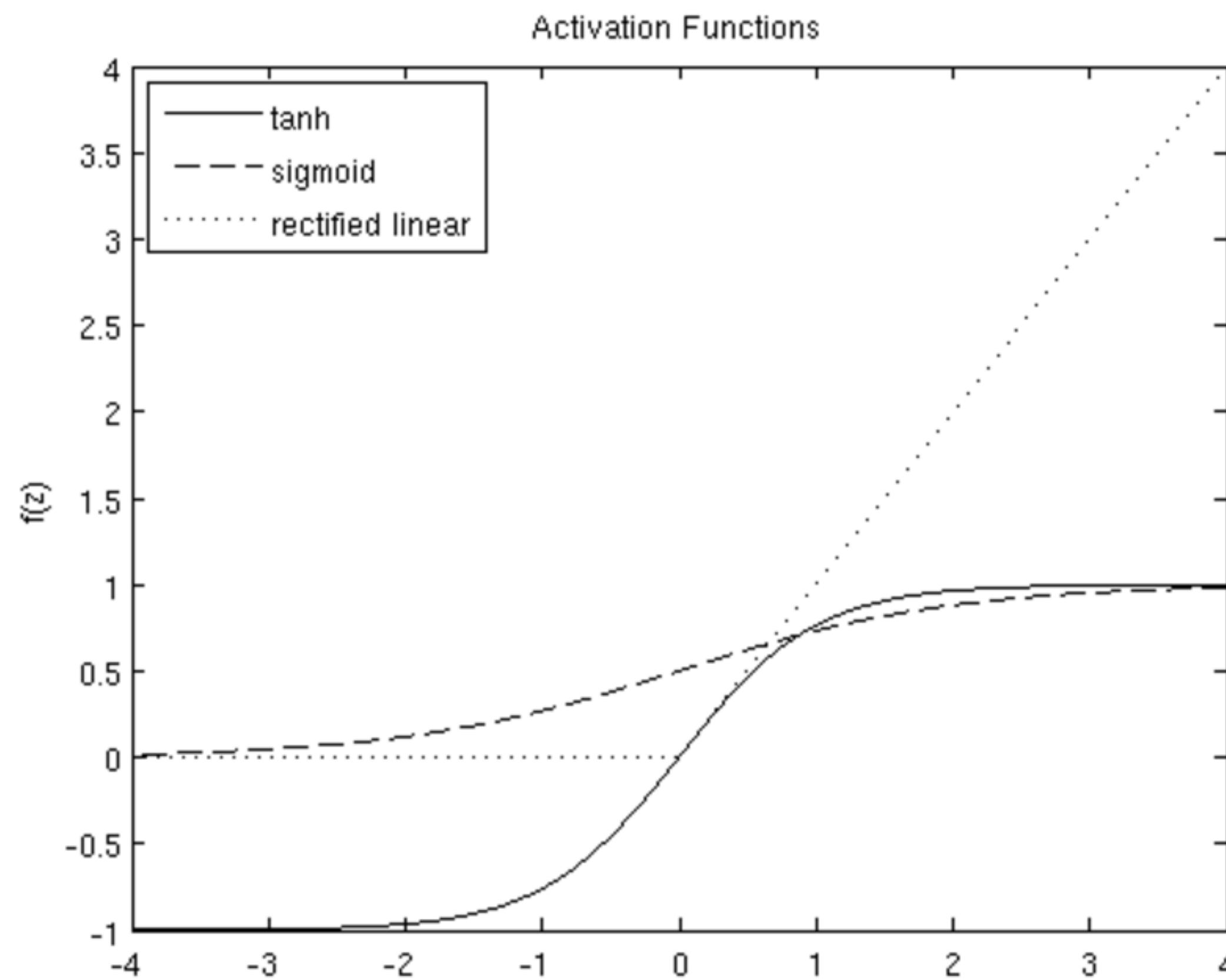
Interesting Question:
What could go wrong with very deep
functions? (3 min)

Neural Network: Definitions



$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^3 W_i x_i + b)$$

Neural Networks: Activation Functions



$$f(z) = \frac{1}{1 + \exp(-z)}$$

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$f(z) = \max(0, z)$$

Neural Networks: Definitions

Feedforward Propagation: Scalar Form

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$

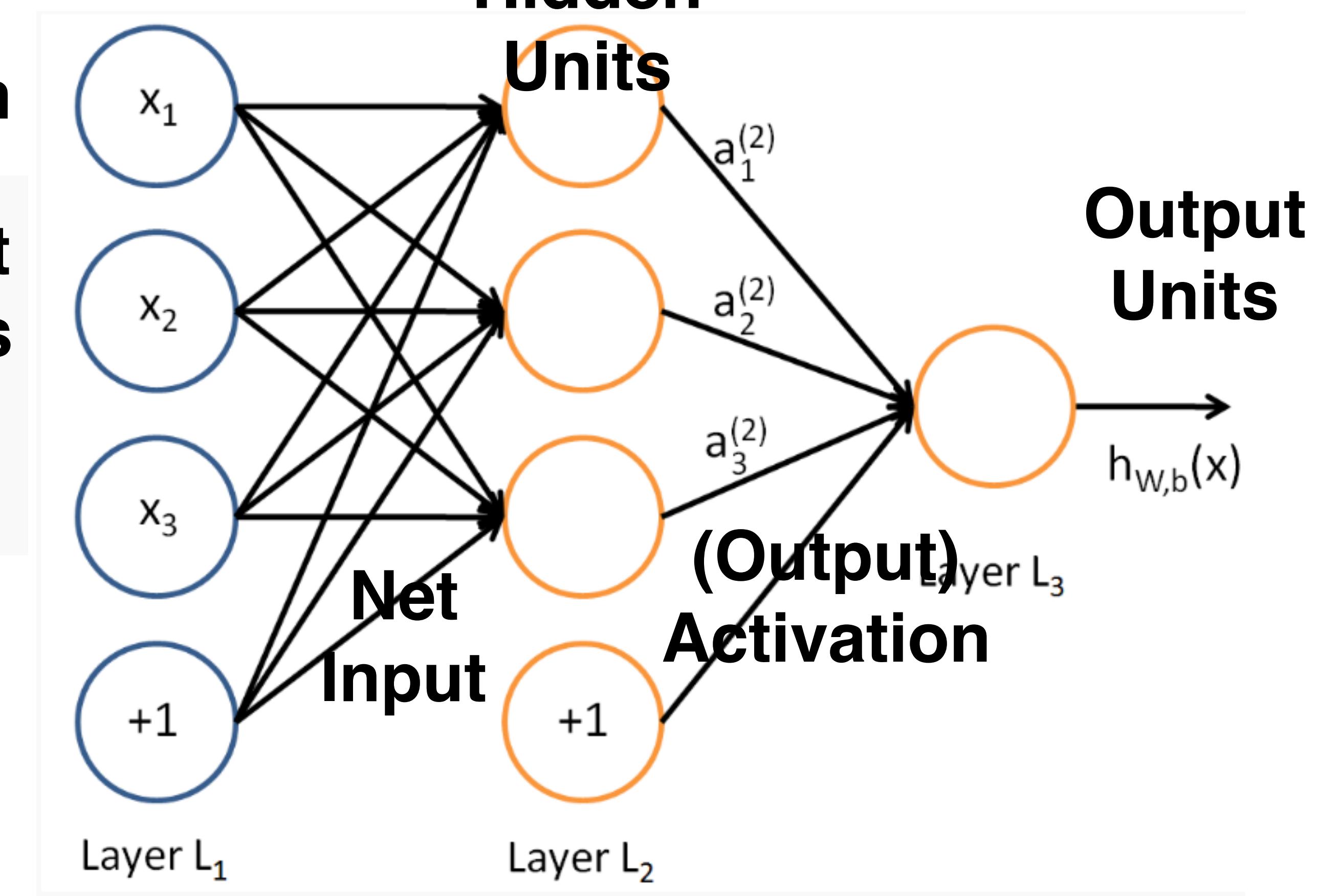
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

$$z_i^{(2)} = \sum_{j=1}^n W_{ij}^{(1)}x_j + b_i^{(1)}$$

Input Units

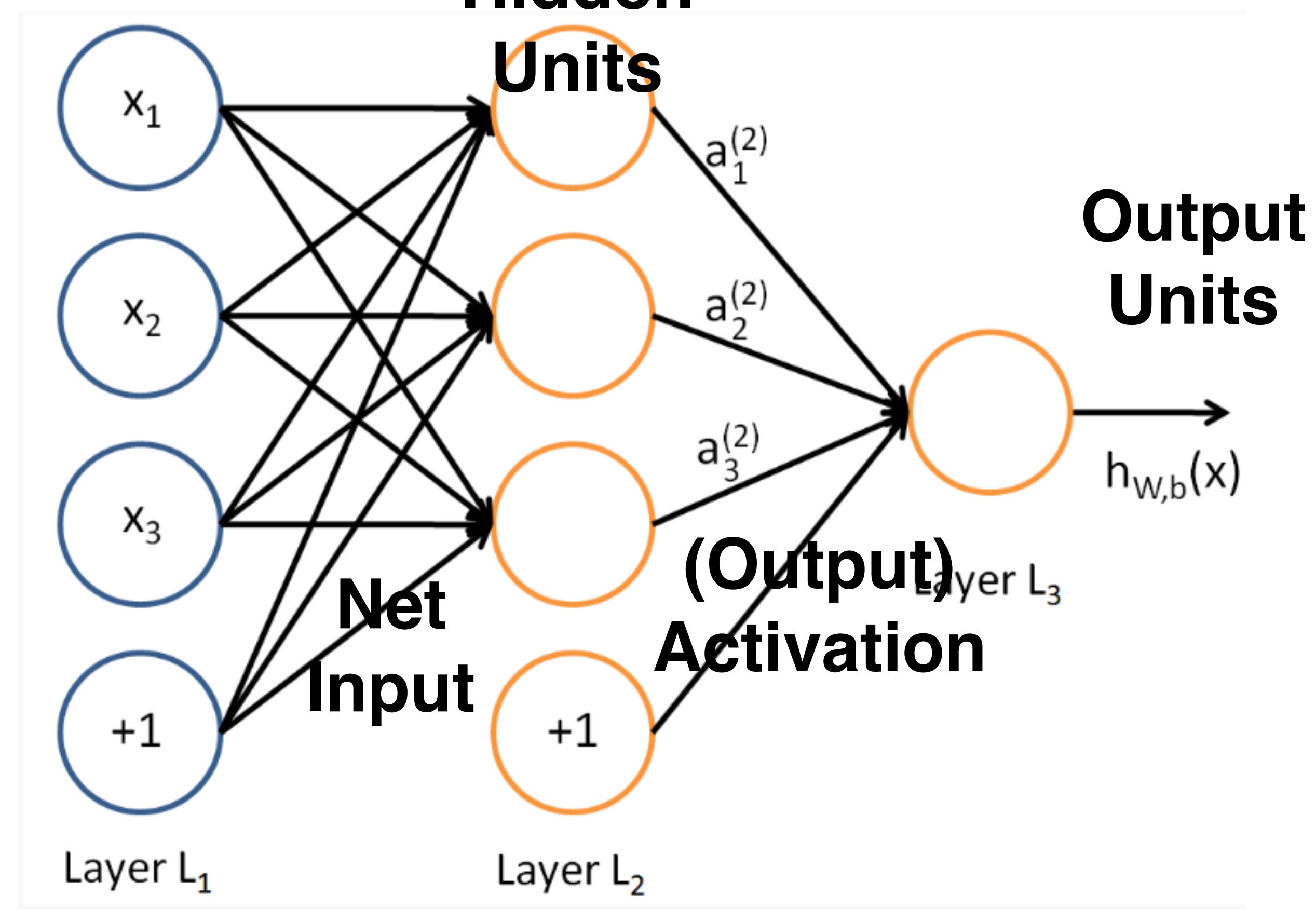


Neural Networks: Definitions

Feedforward Propagation: Vector Form

$$\begin{aligned}z^{(2)} &= W^{(1)}x + b^{(1)} \\a^{(2)} &= f(z^{(2)}) \\z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\h_{W,b}(x) &= a^{(3)} = f(z^{(3)})\end{aligned}$$

Input Units



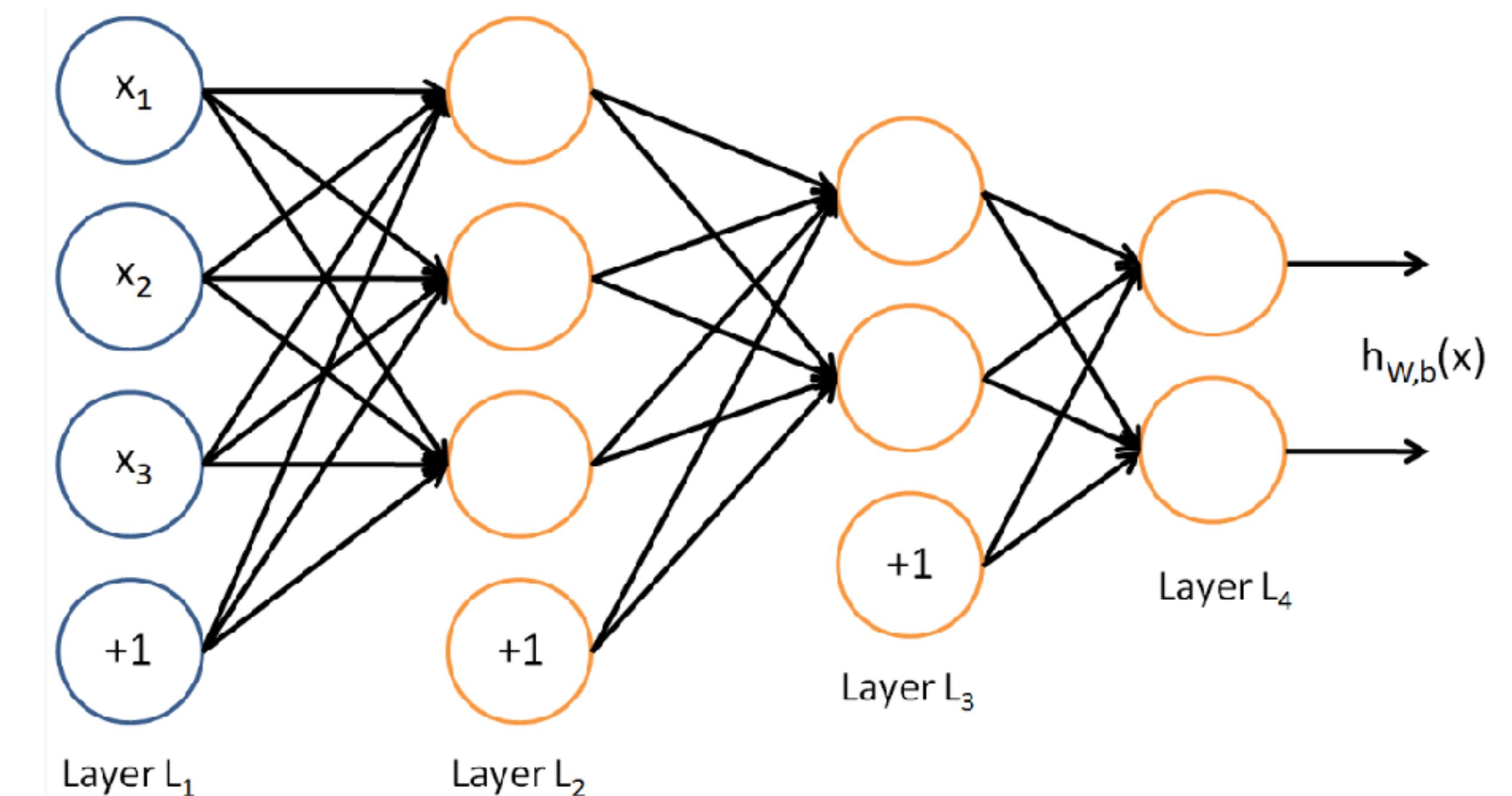
Neural Networks: Definitions

Deep Feedforward Propagation: Vector Form

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)})$$

$$a^{(1)} = x$$

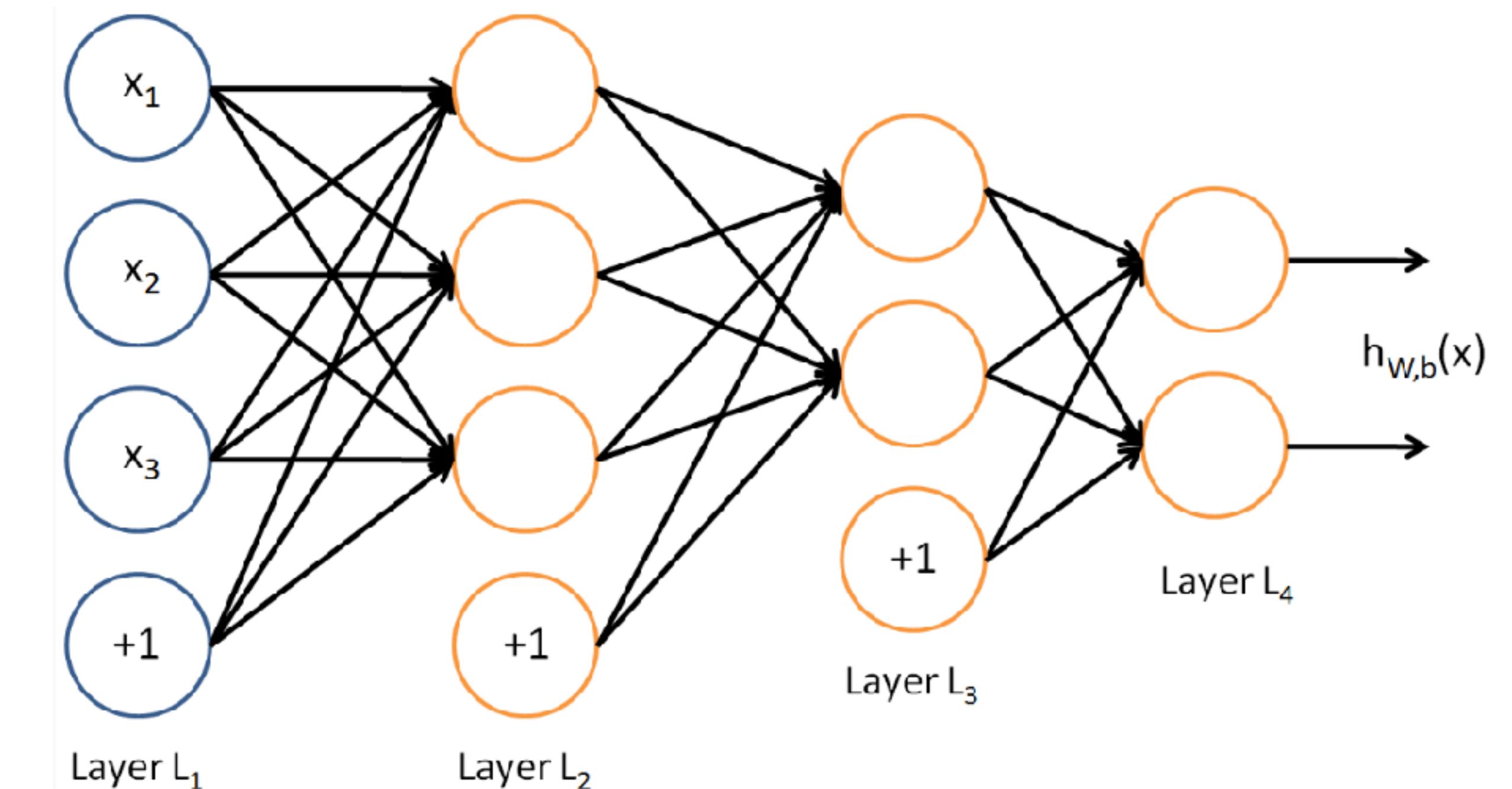


Neural Networks: Definitions

The Training Objective

$$J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right]$$

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2$$



Pseudo-Code for Backprop: Scalar Form

1. Perform a feedforward pass, computing the activations for layers L_2, L_3 , and so on up to the output layer L_{n_l} .
2. For each output unit i in layer n_l (the output layer), set

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

3. For $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$

For each node i in layer l , set

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

4. Compute the desired partial derivatives, which are given as:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}.$$

Pseudo-Code for Backprop: Matrix-Vector Form

1. Perform a feedforward pass, computing the activations for layers L_2, L_3 , up to the output layer L_{n_l} , using the equations defining the forward propagation steps
2. For the output layer (layer n_l), set

$$\delta^{(n_l)} = -(y - a^{(n_l)}) \bullet f'(z^{(n_l)})$$

3. For $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$, set

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) \bullet f'(z^{(l)})$$

4. Compute the desired partial derivatives:

$$\begin{aligned}\nabla_{W^{(l)}} J(W, b; x, y) &= \delta^{(l+1)} (a^{(l)})^T, \\ \nabla_{b^{(l)}} J(W, b; x, y) &= \delta^{(l+1)}.\end{aligned}$$

Gradient Descent for Neural Networks

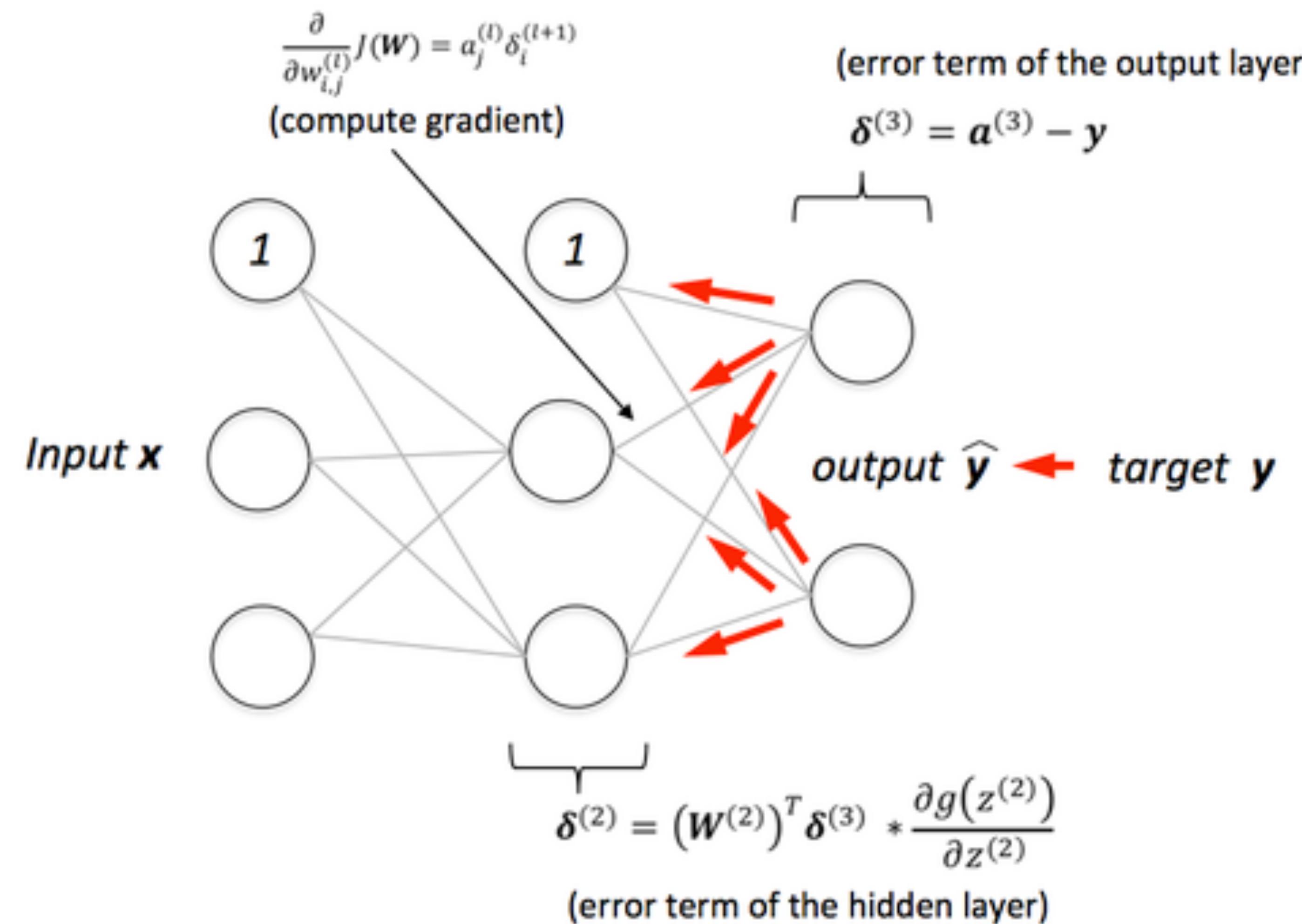
1. Set $\Delta W^{(l)} := 0$, $\Delta b^{(l)} := 0$ (matrix/vector of zeros) for all l .
2. For $i = 1$ to m ,
 1. Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$.
 2. Set $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$.
 3. Set $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$.
3. Update the parameters:

Batch (variance may too large);
try minibatch if too large dataset

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right] \text{ Regularization}$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

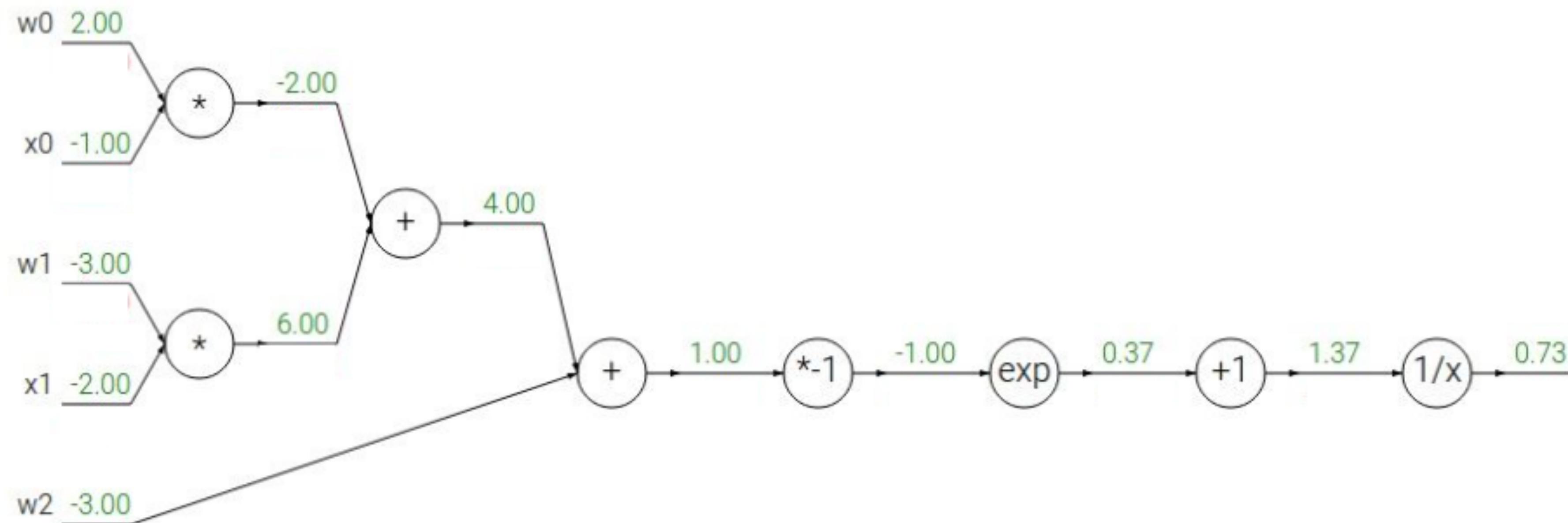
Backpropagation: Network View



Example

Another example:

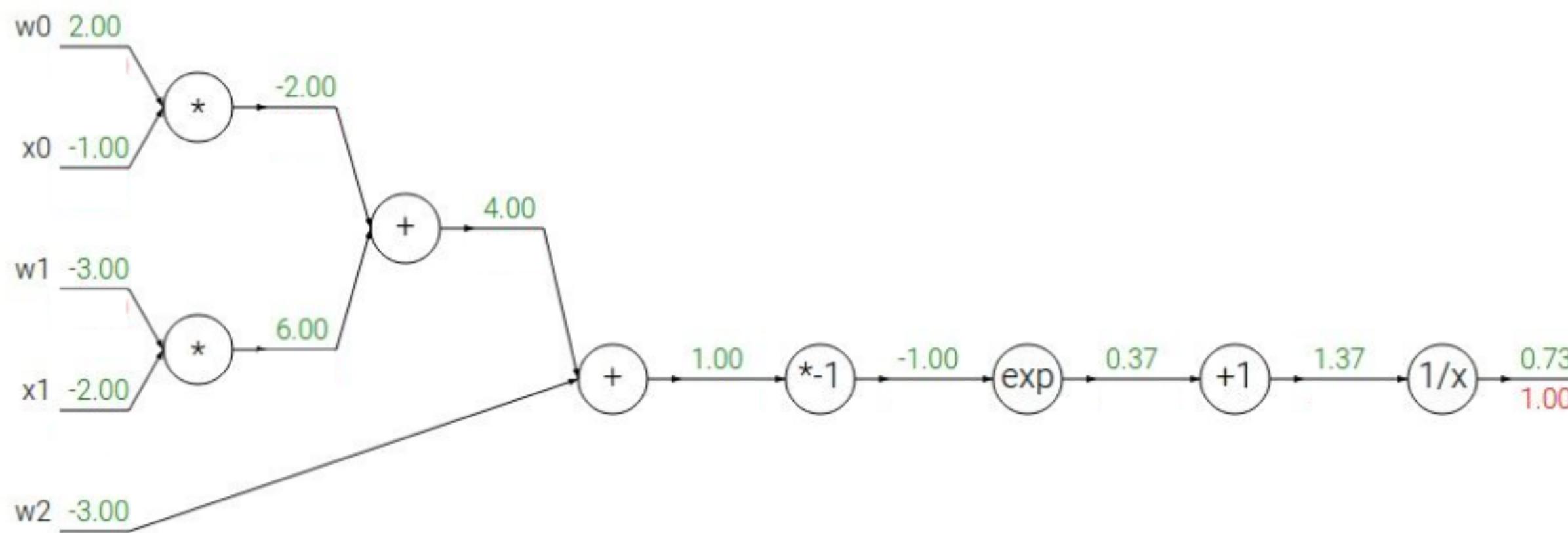
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Example

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

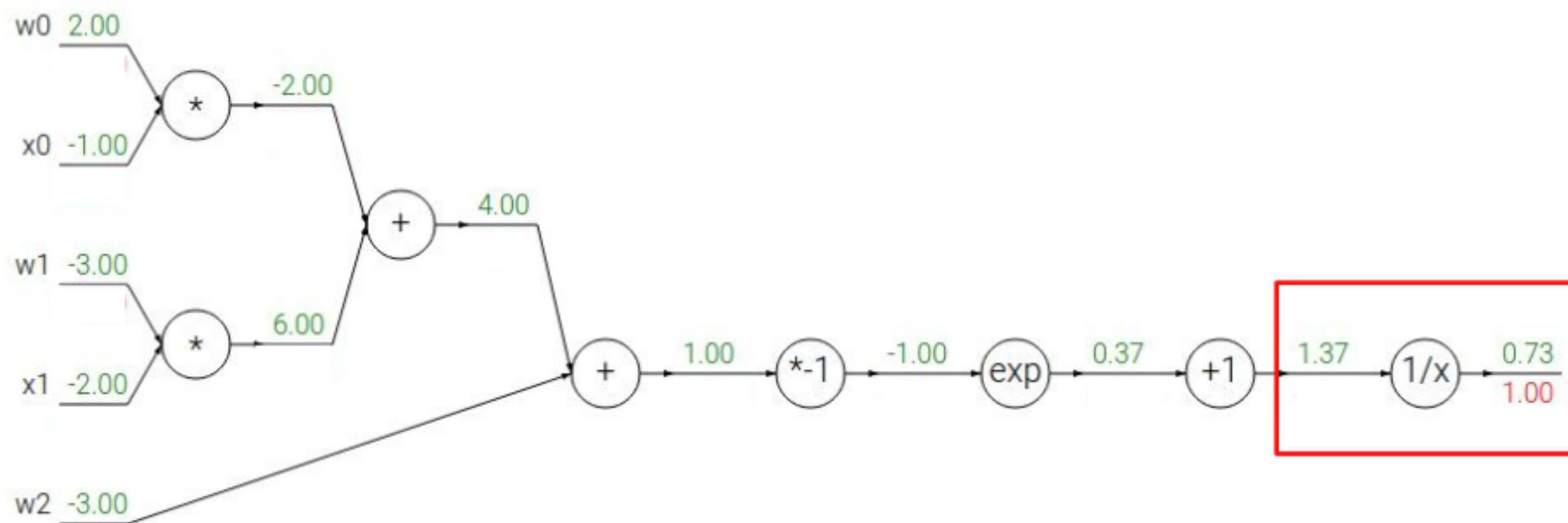
→

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

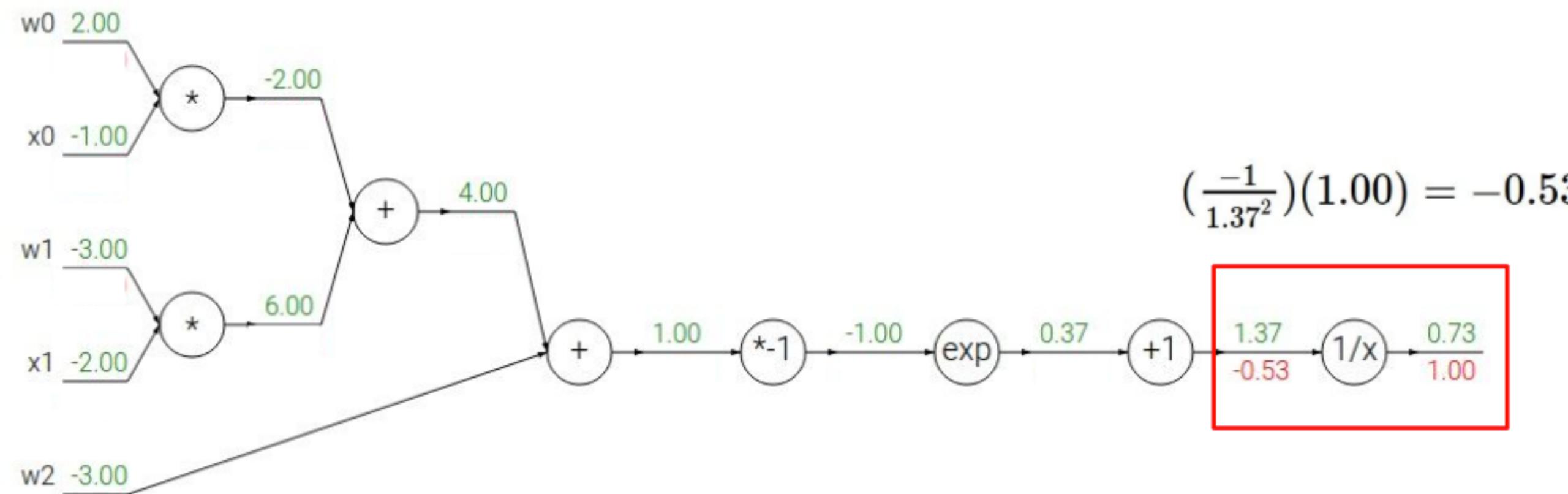
→

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

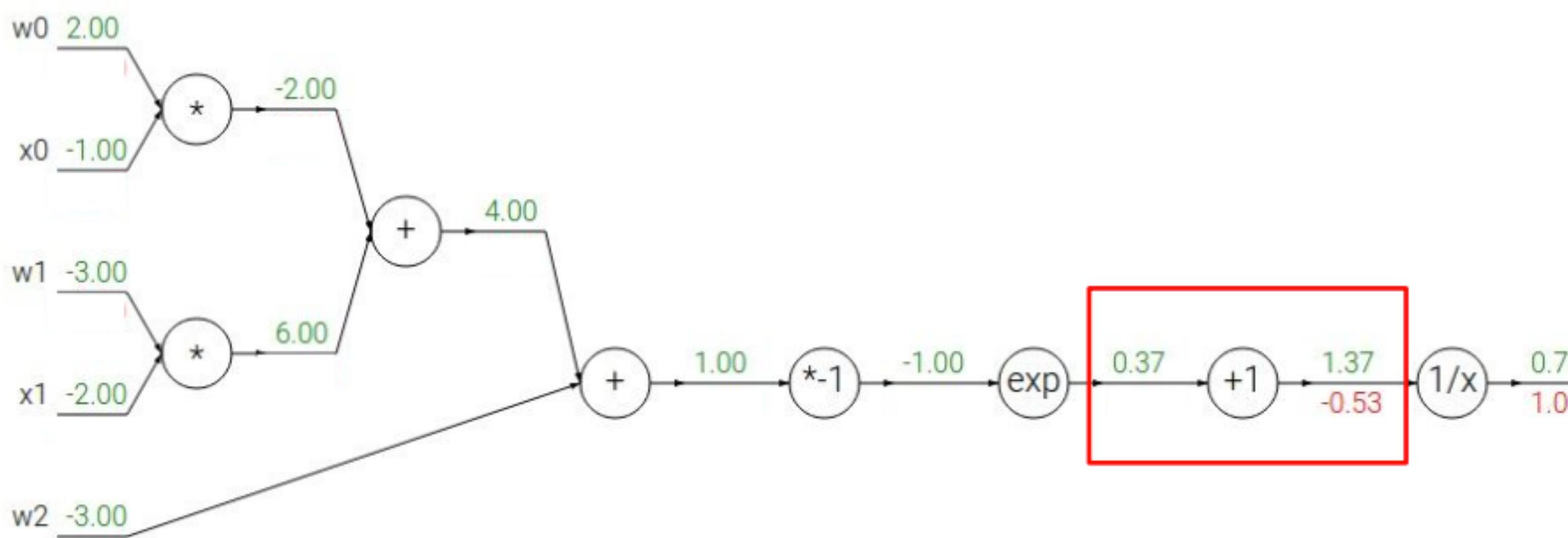
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

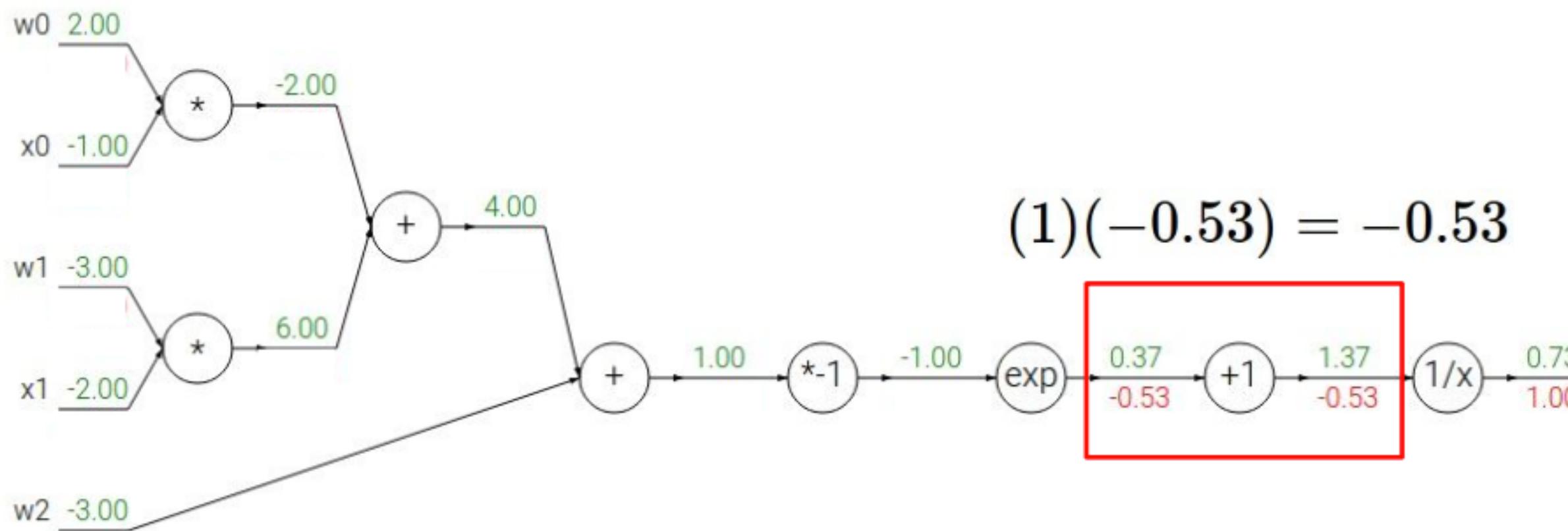
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

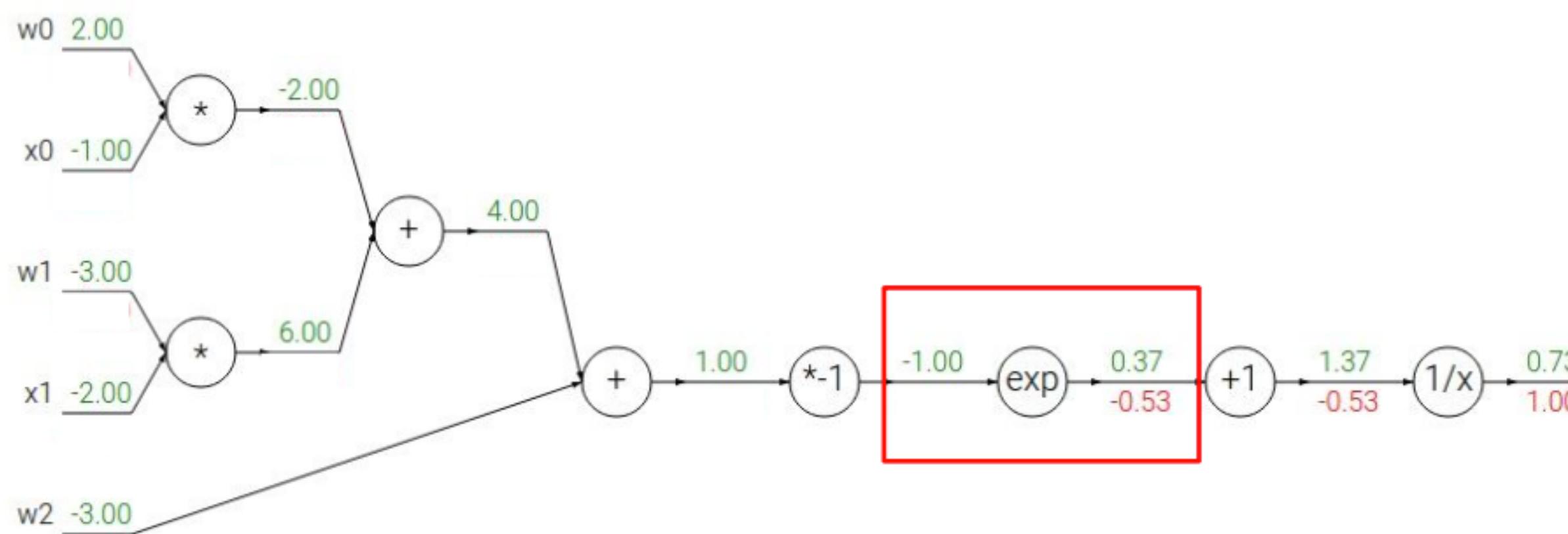
→

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

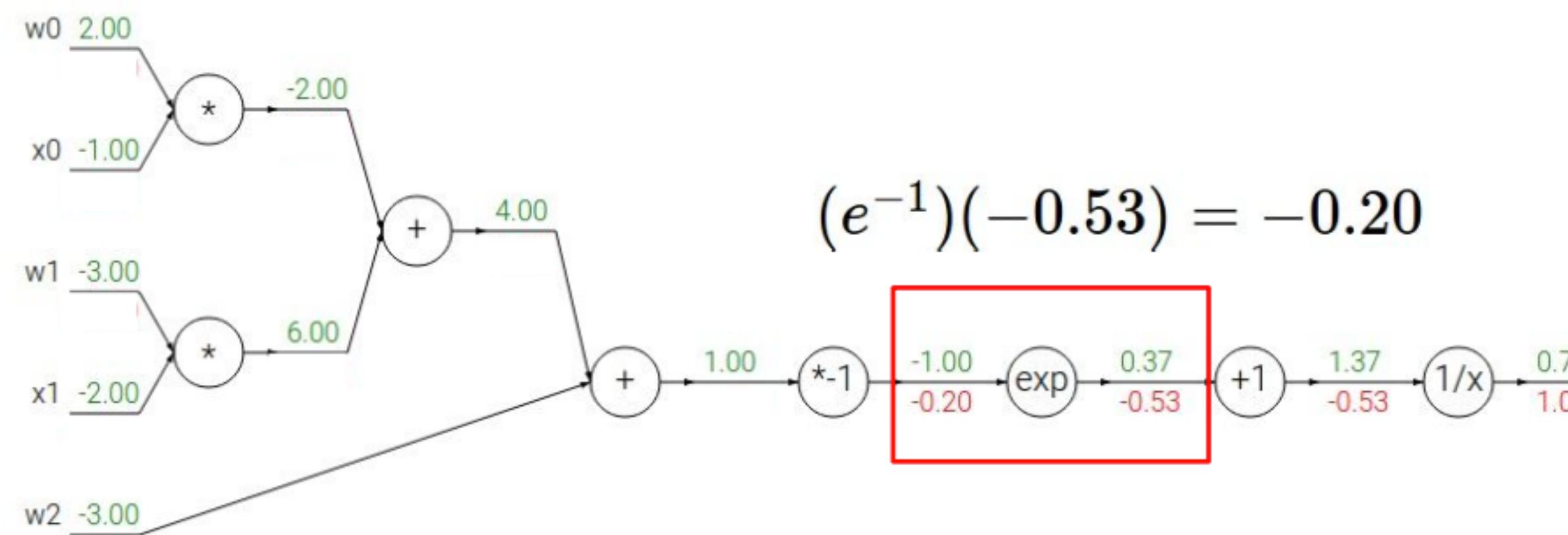
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



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

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

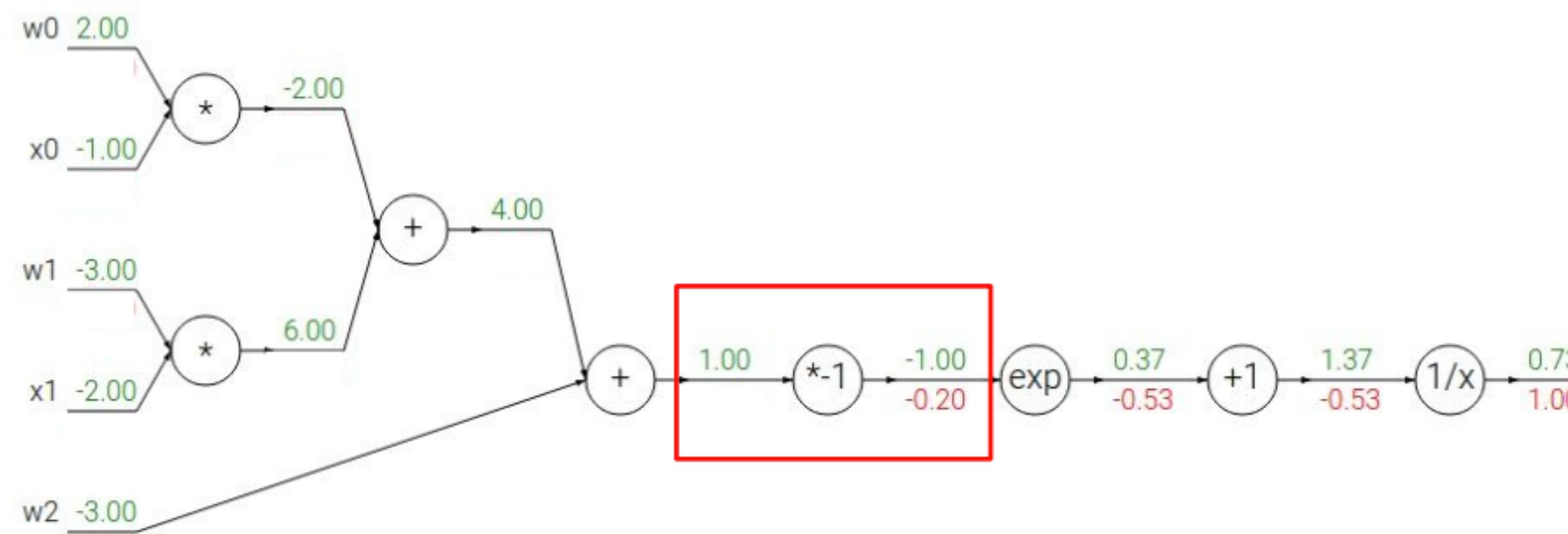
$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

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

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

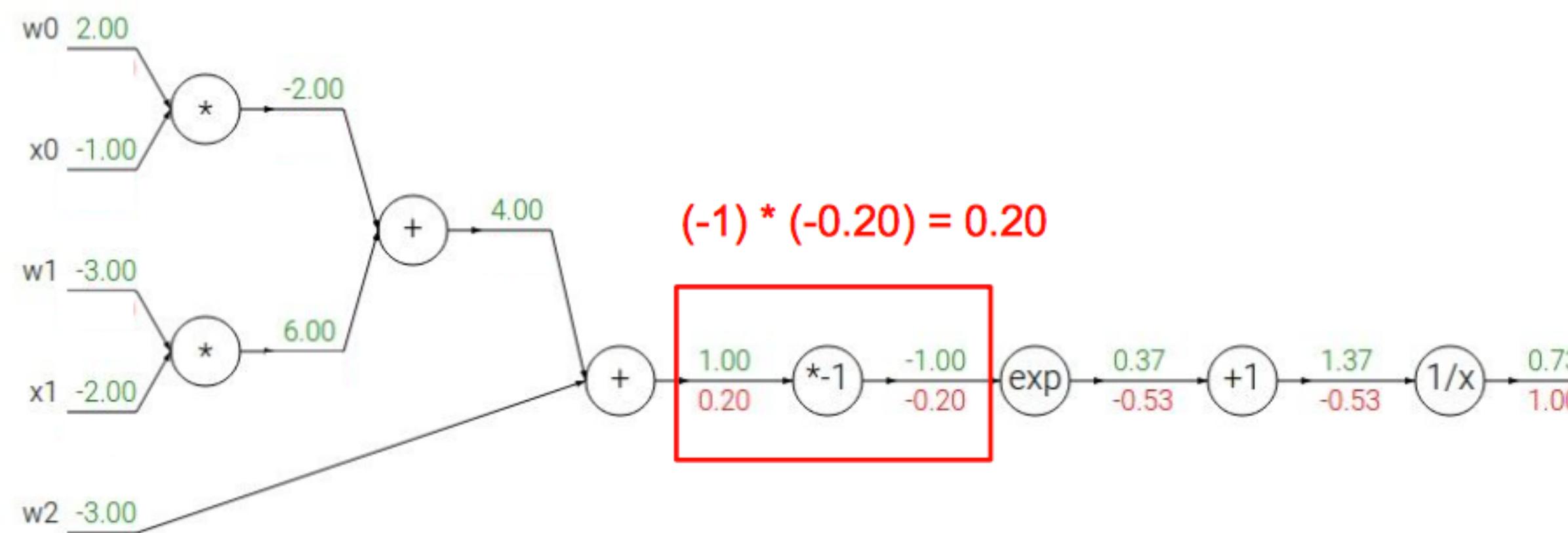
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

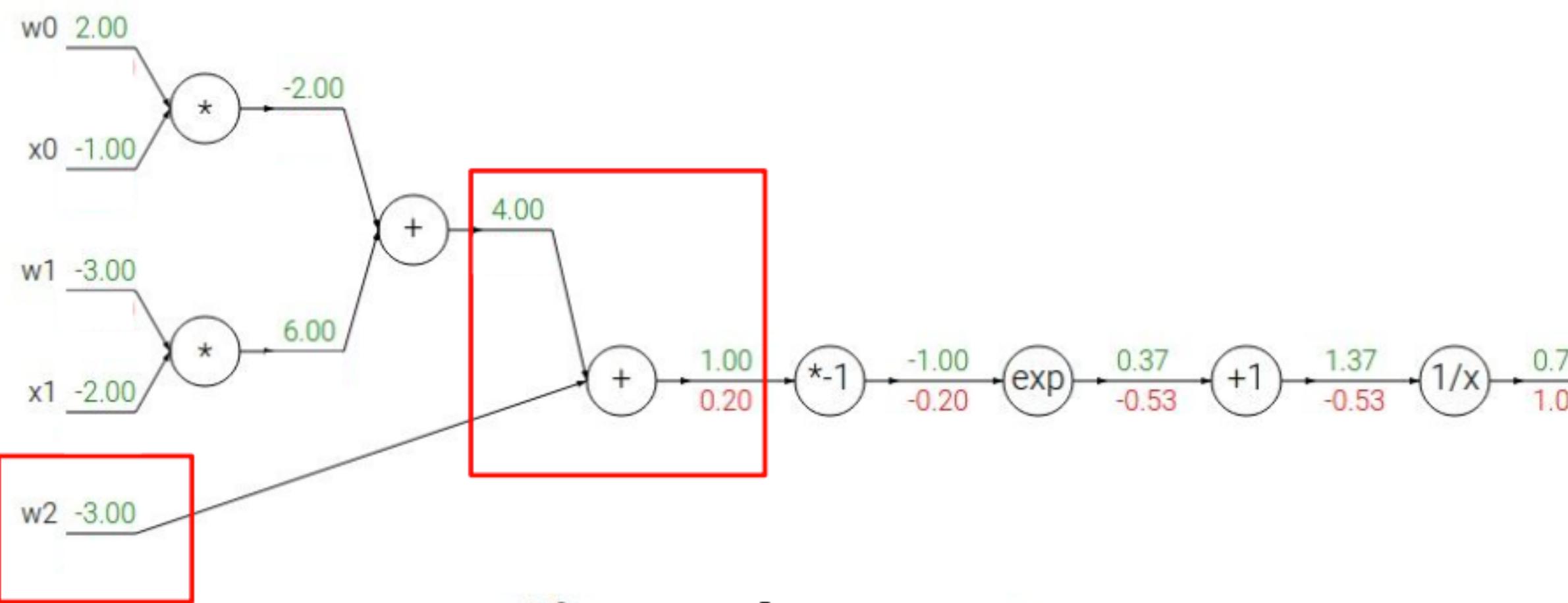
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

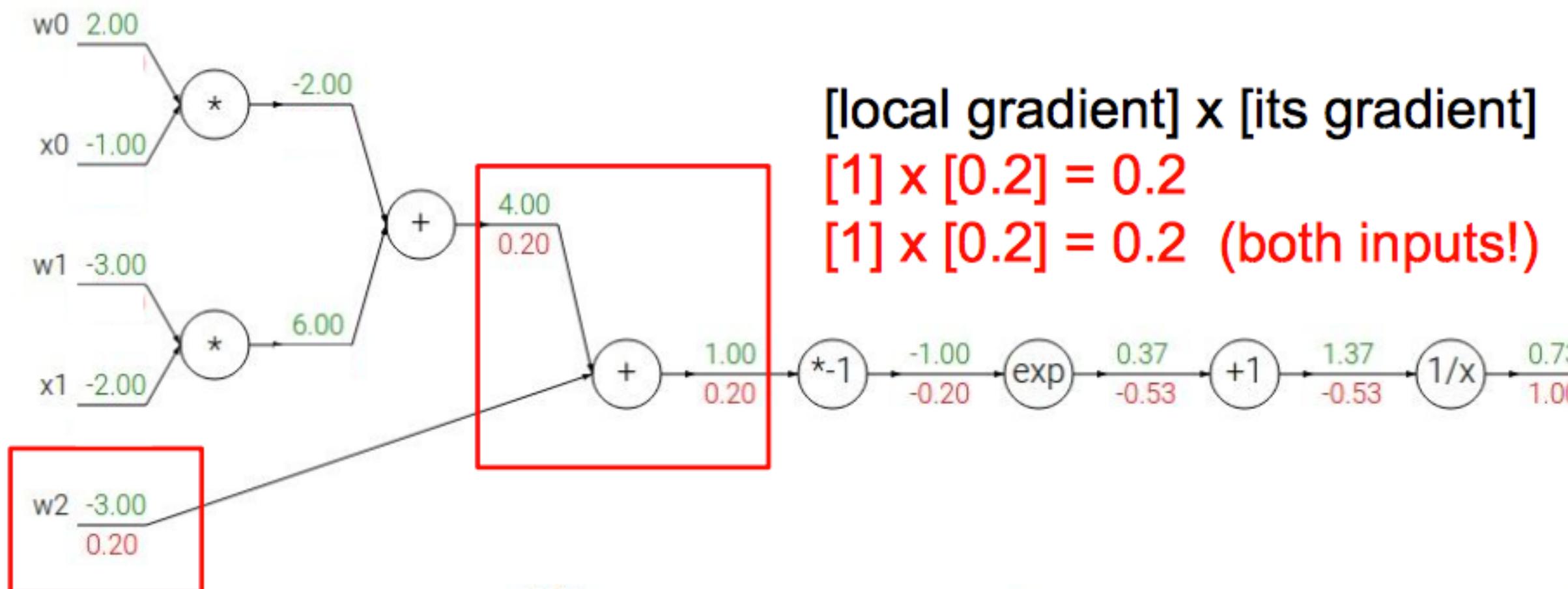
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

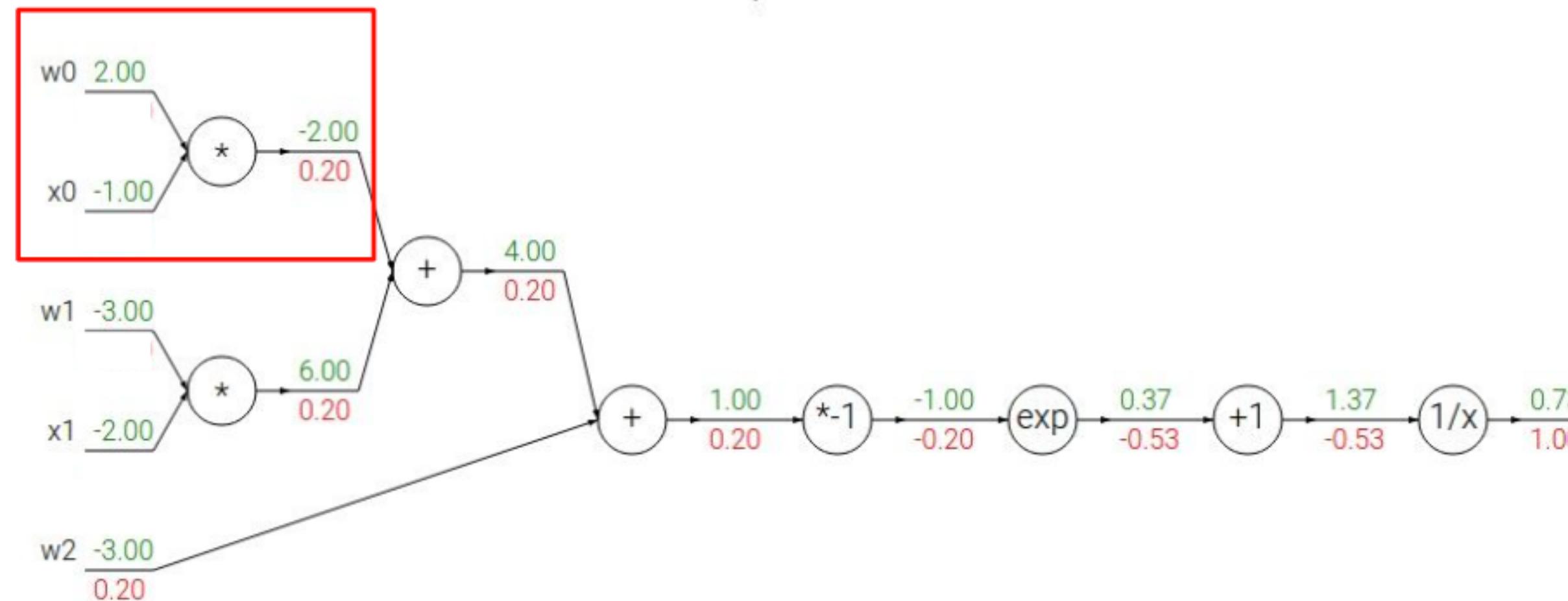
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

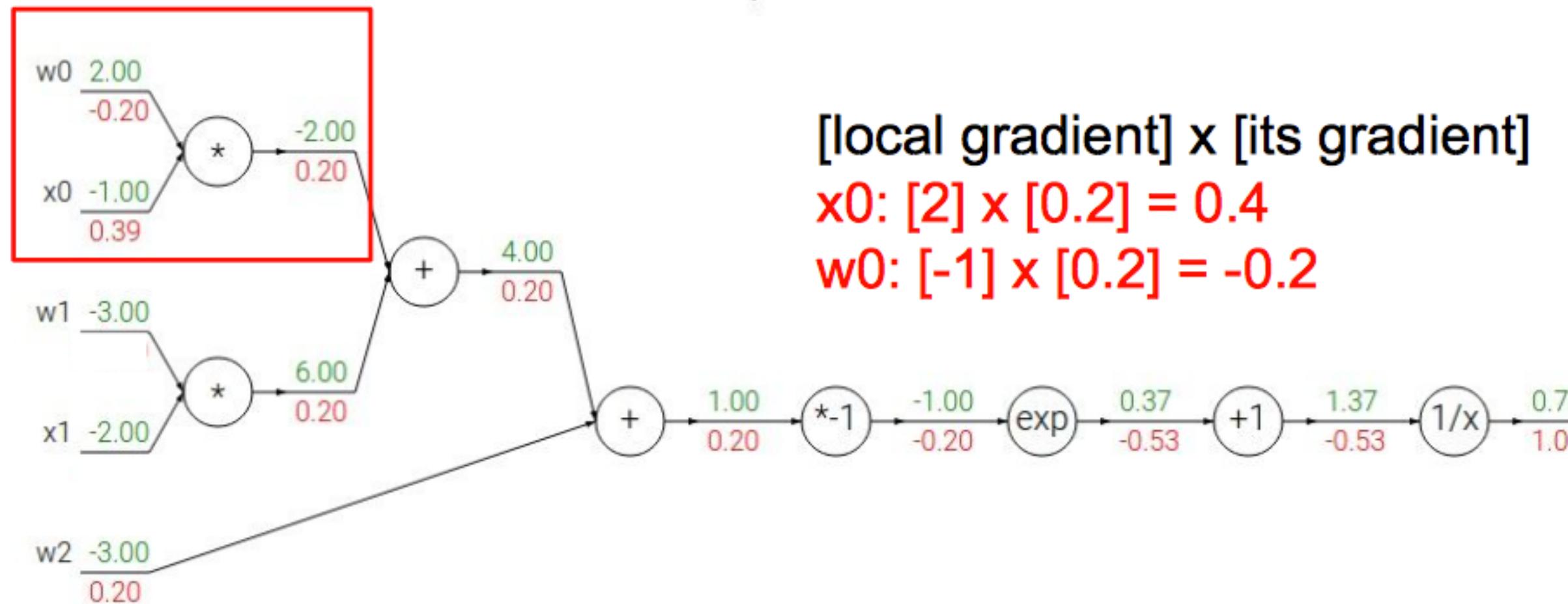
\rightarrow

$$\frac{df}{dx} = 1$$

Example

Another example:

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



[local gradient] \times [its gradient]

$$x_0: [2] \times [0.2] = 0.4$$

$$w_0: [-1] \times [0.2] = -0.2$$

$$f(x) = e^x$$

\rightarrow

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

\rightarrow

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

\rightarrow

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

\rightarrow

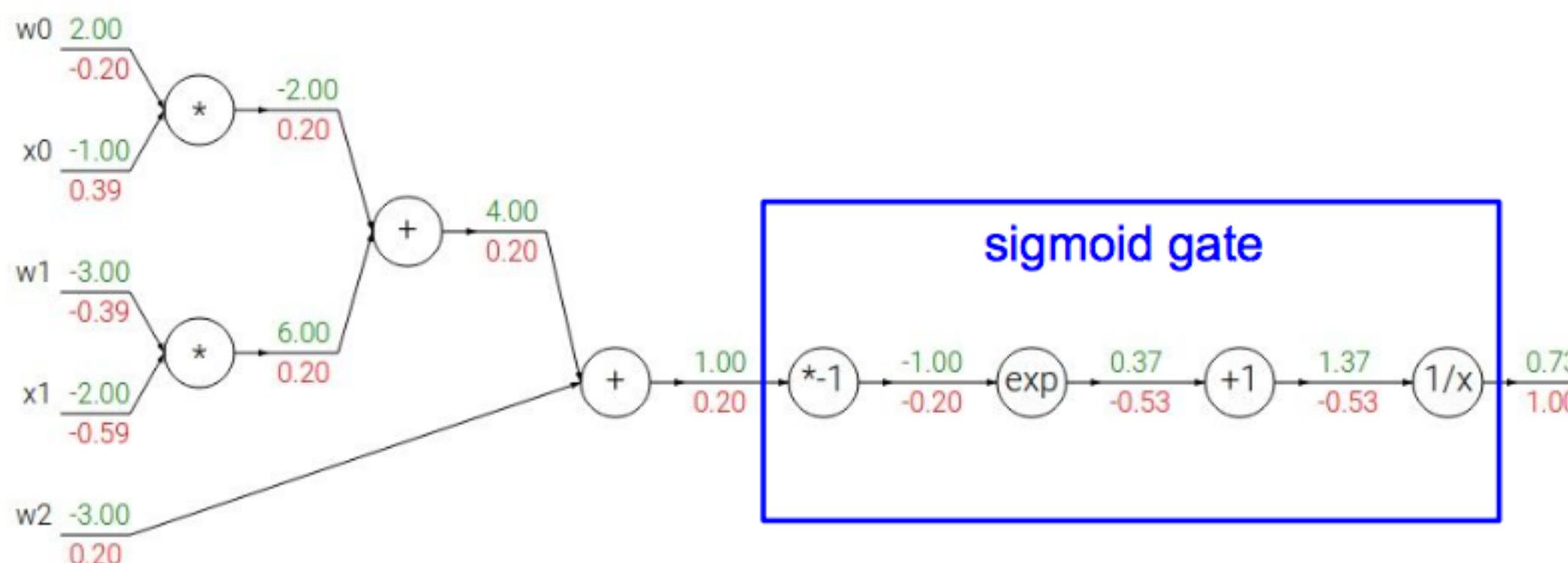
$$\frac{df}{dx} = 1$$

Example

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

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

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



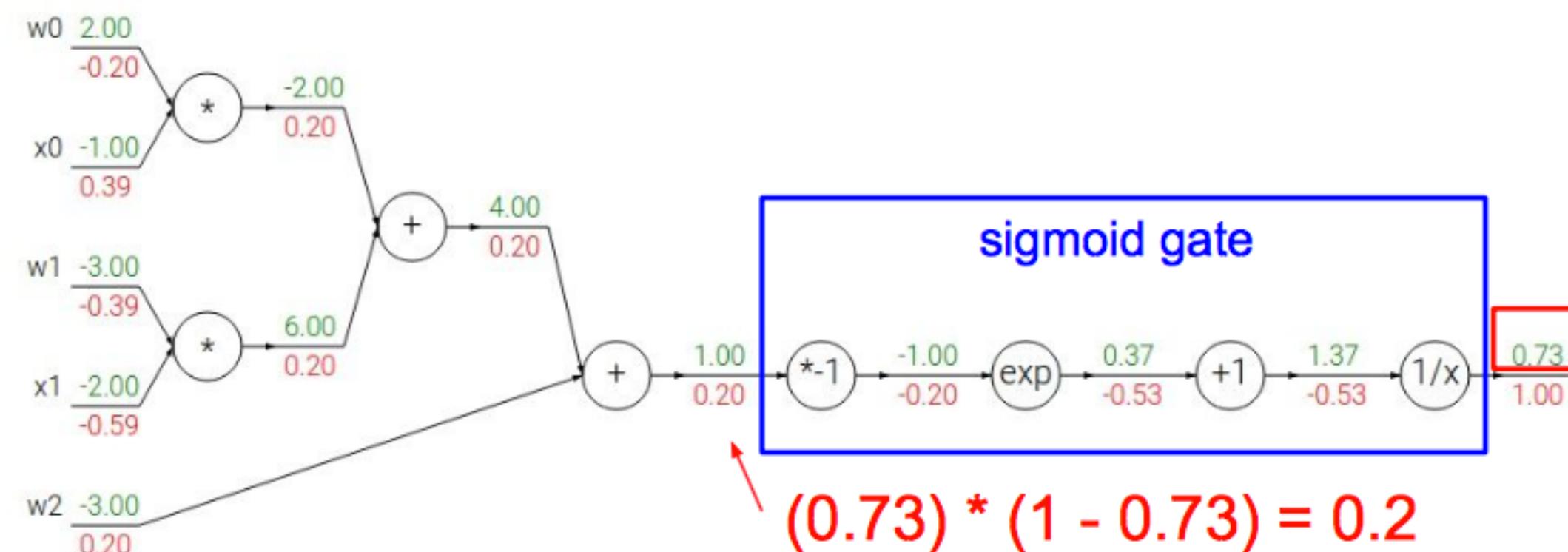
[Fei-Fei Li, Andrej Karpathy, Justin Johnson]

Example

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

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

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



$$(0.73) * (1 - 0.73) = 0.2$$

[Fei-Fei Li, Andrej Karpathy, Justin Johnson]

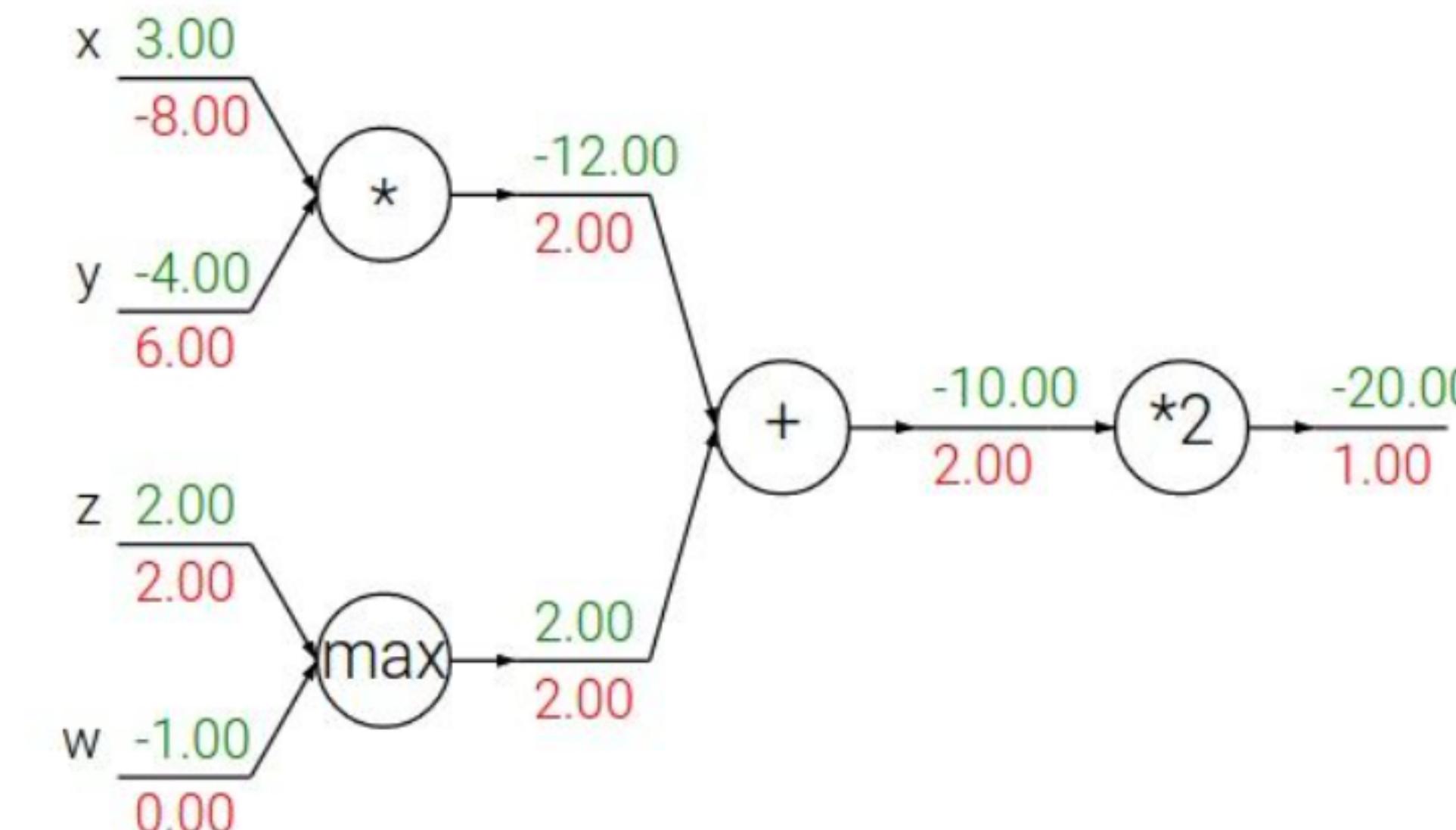
Example

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

mul gate: gradient... “switcher”?



[Fei-Fei Li, Andrej Karpathy, Justin Johnson]

Classification with 2-Layer Neural Network

- [http://cs.stanford.edu/people/karpathy/convnetjs/demo/
classify2d.html](http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html)