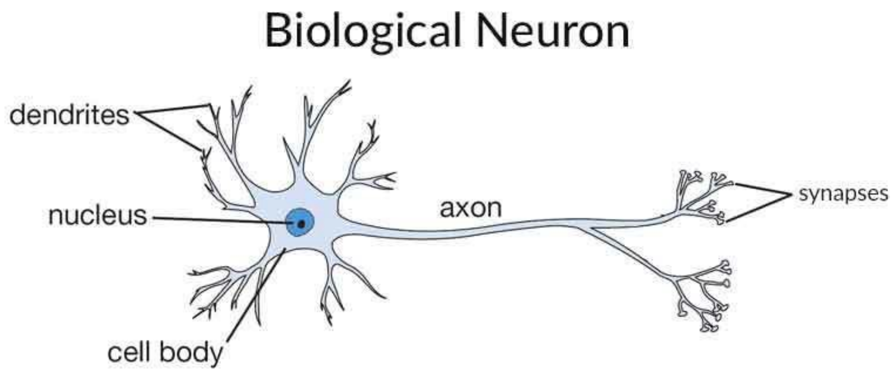
Data Mining

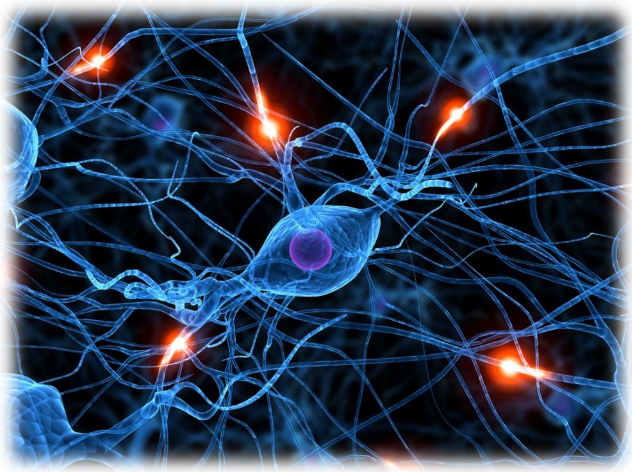
Classification – Neural Networks

Introduction

Deep Learning is a subset of Machine Learning is the human brain embedded in a machine.

It is inspired by the working of a human brain and therefore is a set of neural network algorithms which tries to mimics the working of a human brain and learn from the experiences.





Artificial Neural Networks (ANN)

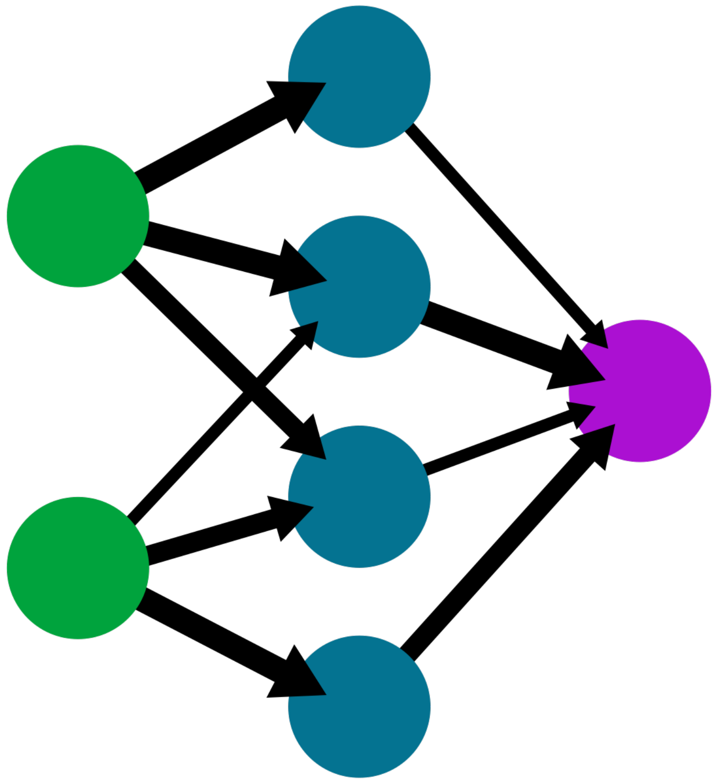
Neural Networks is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.

The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

Artificial Neural Networks (ANN)

A simple neural network consists of three components:

1. Input Layer
2. Hidden Layer
3. Output Layer



Artificial Neural Networks (ANN)

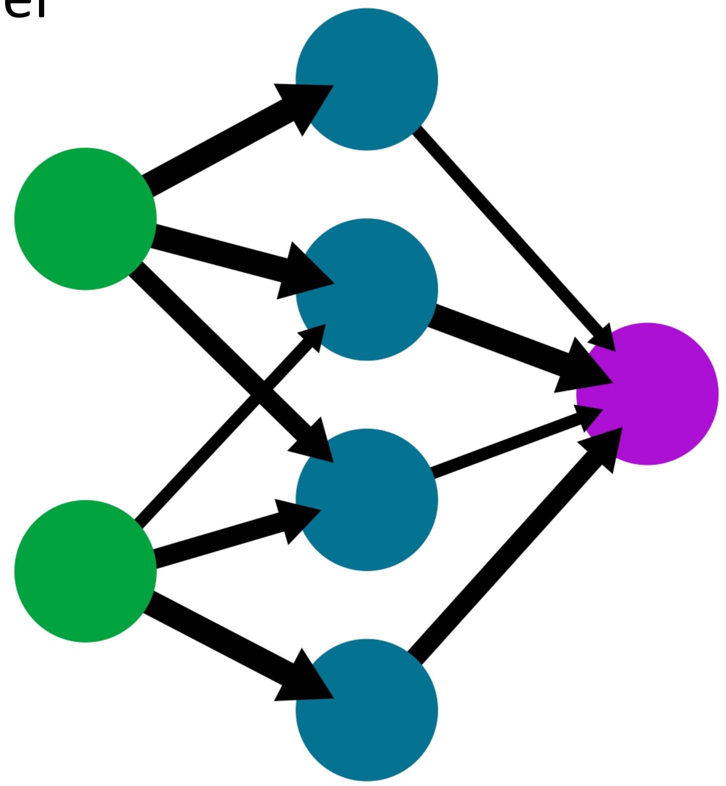
Input Layer: Also known as Input nodes are the inputs/information

from the outside world is provided to the model

to learn and derive conclusions from.

Input nodes pass the information

to the next layer i.e Hidden layer.



Artificial Neural Networks (ANN)

Hidden Layer: Hidden layer is the set of neurons where all

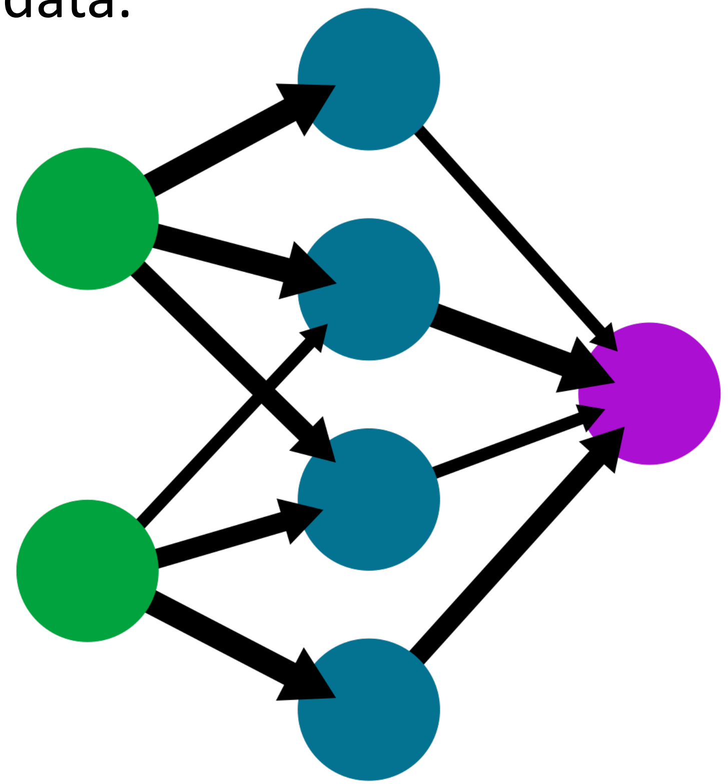
the computations are performed on the input data.

There can be any number of

hidden layers in a neural network.

The simplest network consists

of a single hidden layer.



Artificial Neural Networks (ANN)

Output layer: The output layer is the output/conclusions of the model

derived from all the computations performed.

There can be single or multiple nodes

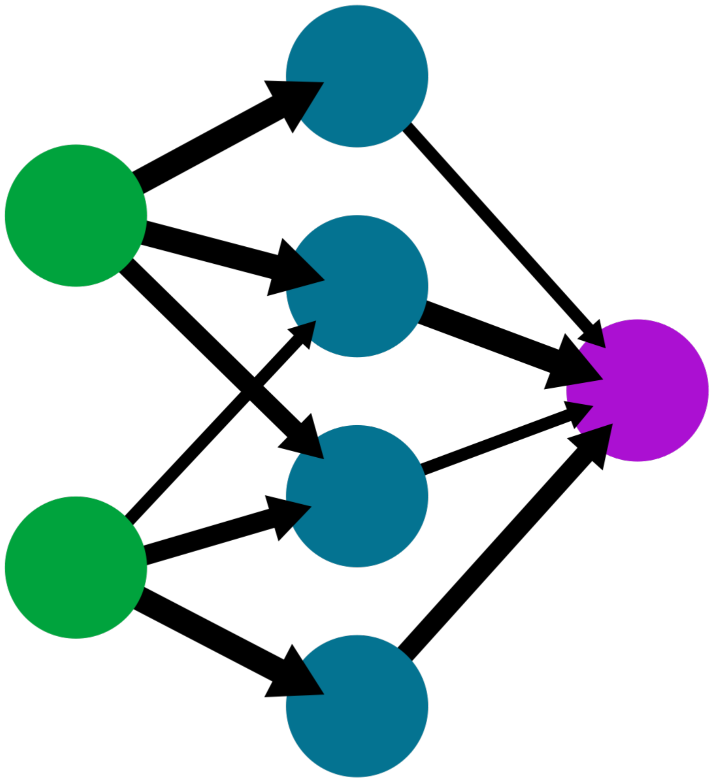
in the output layer. If we have a

binary classification problem the output

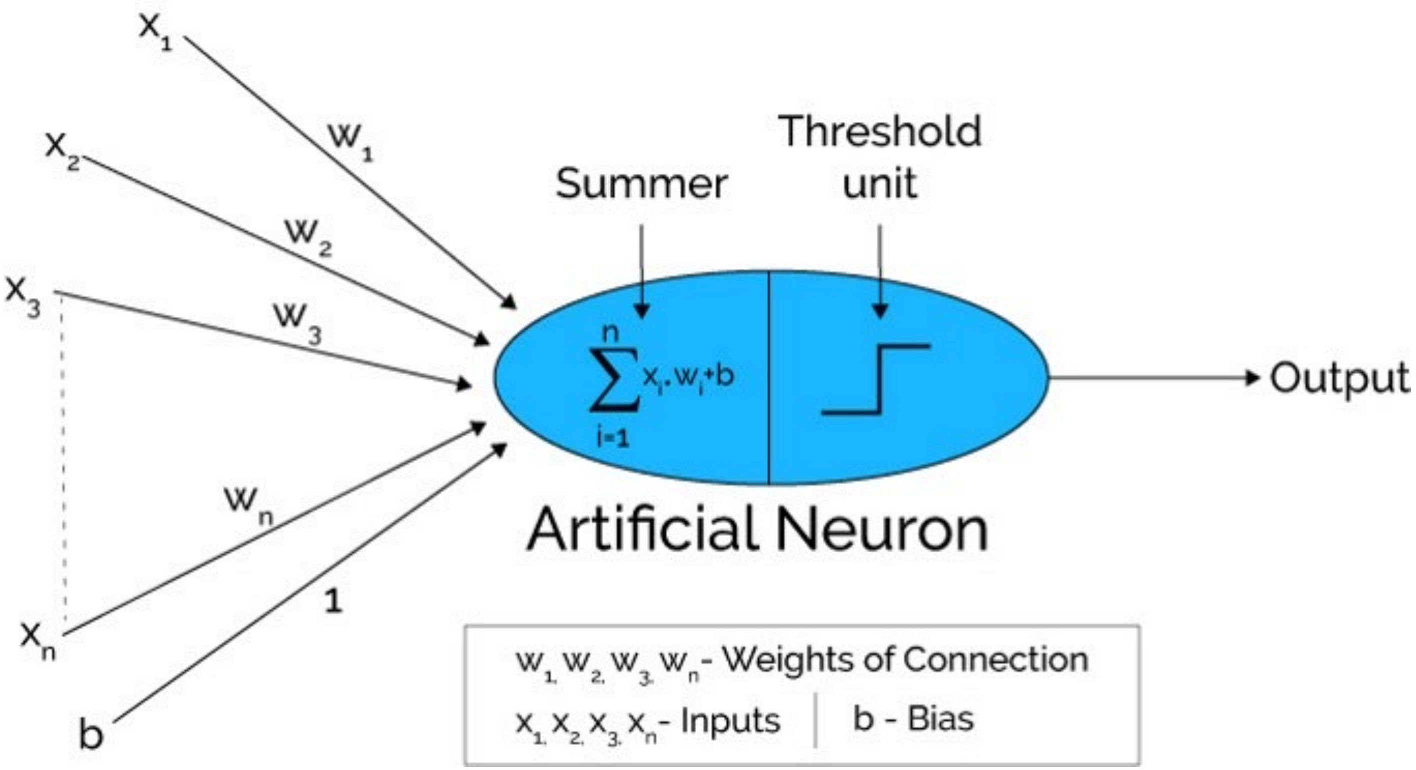
node is 1 but in the case of

multi-class classification,

the output nodes can be more than 1.



Working with NN

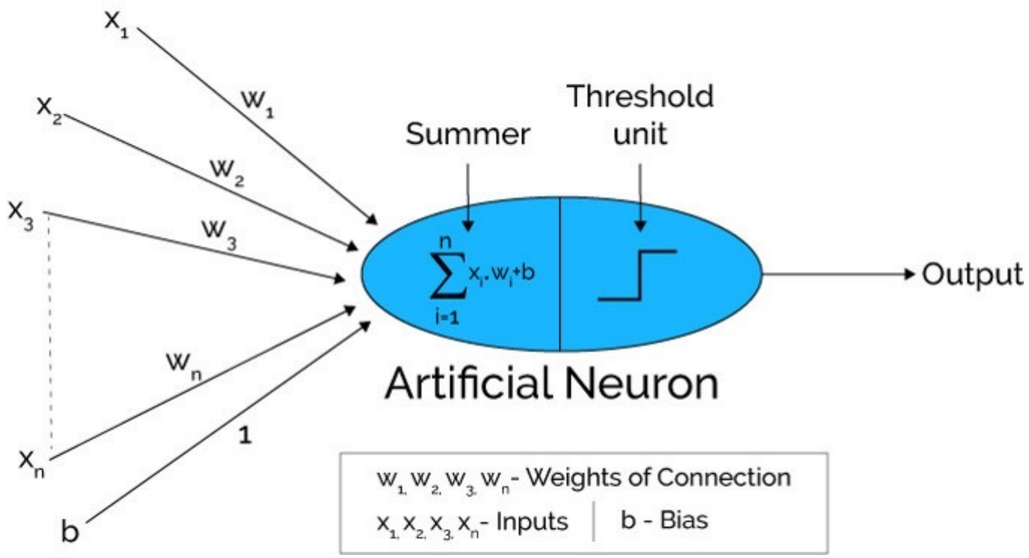


Working with NN

In the first step, **Input units are passed with weights attached to it to the hidden layer**.

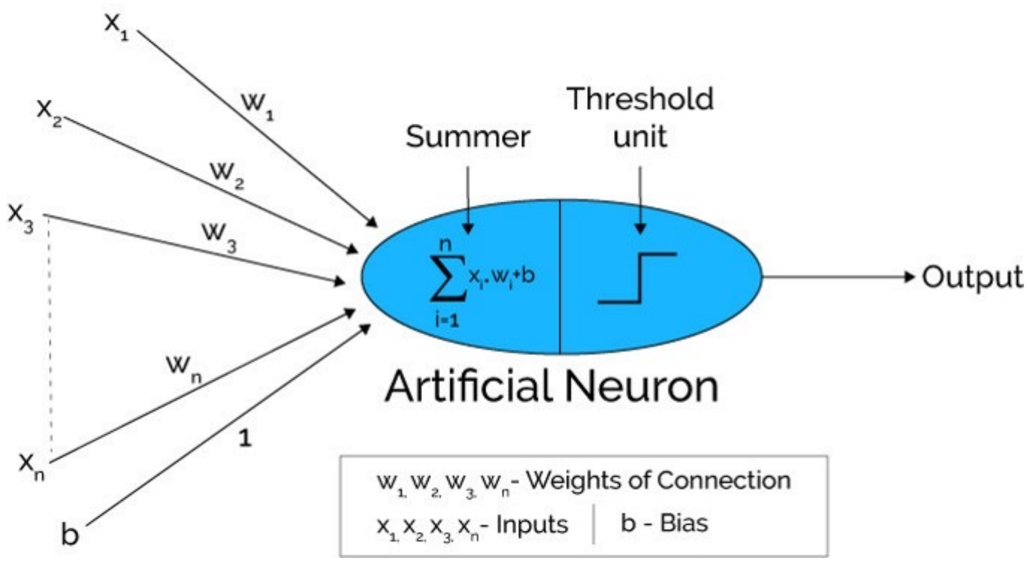
We can have any number of hidden layers. In the image inputs $x_1, x_2, x_3, \dots, x_n$ is passed.

Each hidden layer consists of neurons. All the inputs are connected to each neuron.



Working with NN

After passing on the inputs, **all the computation is performed in the hidden layer** (Blue oval in the picture)



Computation in Hidden Layers

First of all, all the inputs are multiplied by their weights. **It shows the strength of the particular input.**

After assigning the weights, a bias variable is added:

$$Z = w_1 x_1 + \cdots + w_n x_n + b$$

$$OR$$

$$Z = w^T x + b$$

Then in the second step, the **activation function is applied to the linear equation Z. (TBD)**

Computation in Hidden Layers

- The whole process is performed in each hidden layer.
- After passing through every hidden layer, **we move to the last layer (output) which gives us the final output.**
- The process explained above is known as **forwarding Propagation.**
- After getting the predictions from the output layer, the **error is calculated.**
- If the error is large, then the steps are taken to minimize the error and for the same purpose, **Back Propagation is performed.**

What is Back Propagation?

Back Propagation is the process of updating and finding the optimal values of weights or coefficients which helps the model to minimize the error i.e difference between the actual and predicted values.

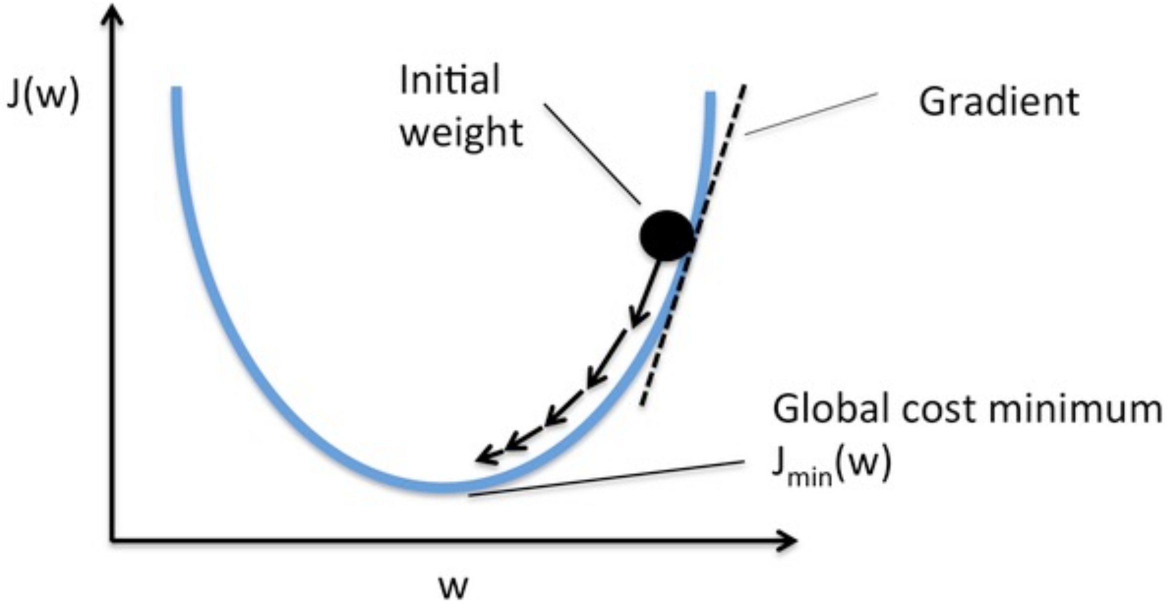
How the weights are updated and new weights are calculated?

The weights are updated with the help of optimizers.

Back Propagation with Gradient Descent

Gradient Descent is one of the optimizers which helps in calculating the new weights.

In the image below, the curve is our cost function curve, and our aim is the minimize the error such that J_{min} i.e global minima is achieved.



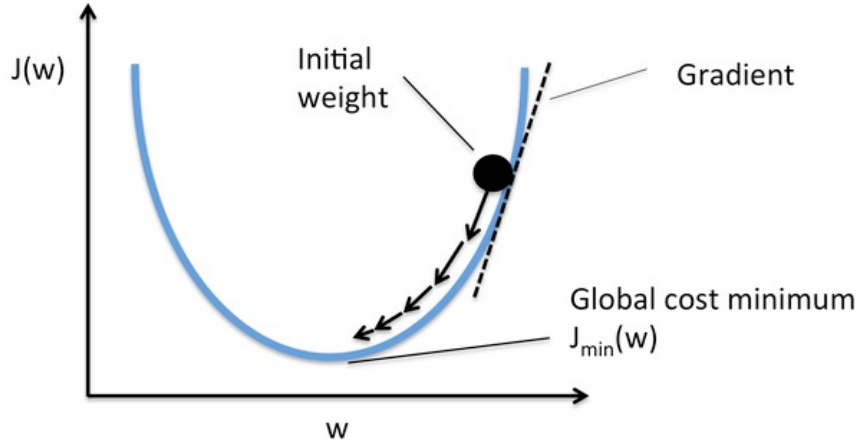
Back Propagation with Gradient Descent

W_x^* - the new weight

W_x - the old weight

α - learning rate

Each step, calculate:



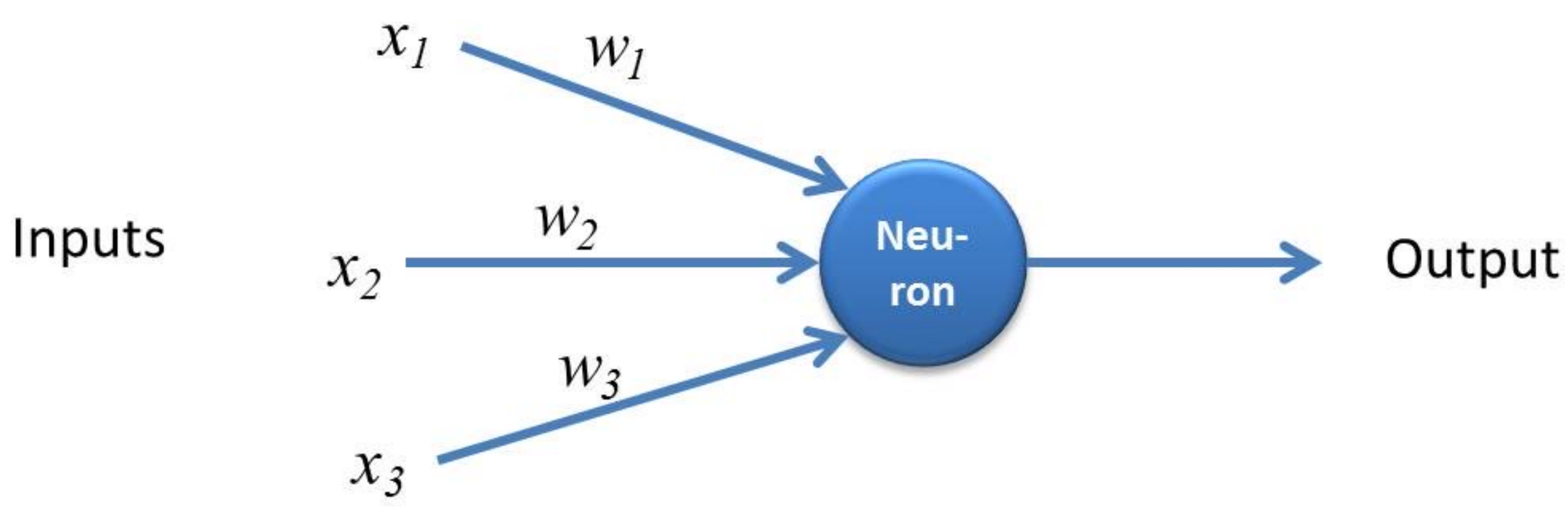
$$W_x^* = W_x - \alpha \cdot \frac{\partial Error}{\partial W_x}$$

This process of calculating the new weights, then errors from the new weights, and then updation of weights **continues till we reach global minima and loss is minimized**.

Perceptron

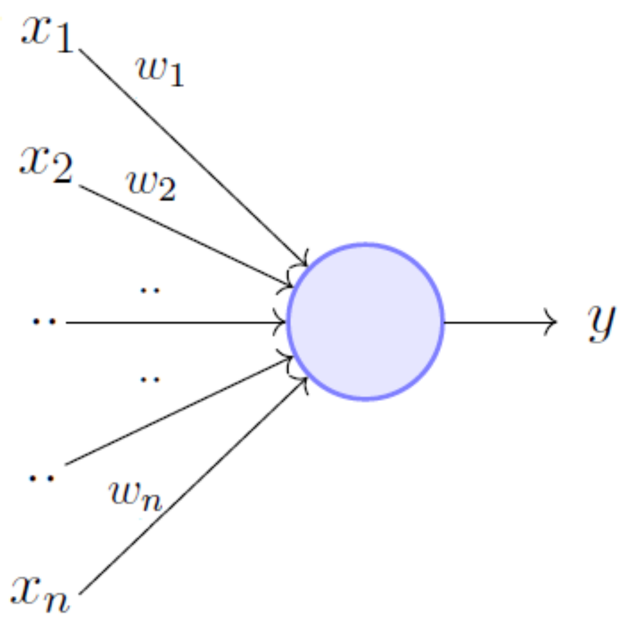
Perceptron

Perceptron is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.



Perceptron

It takes an input, aggregates it (weighted sum) and returns 1 only if the aggregated sum is more than some threshold else returns 0.



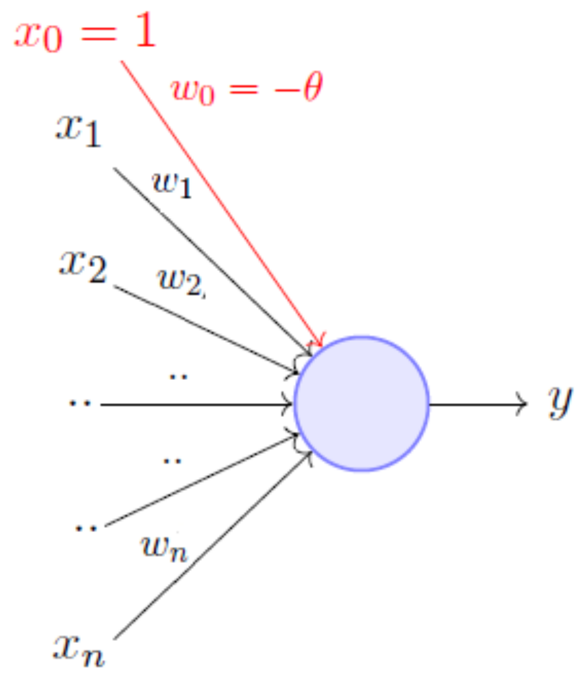
$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

Perceptron

Rewriting the threshold as shown above and making it a constant input with a variable weight, we would end up with something like the following:



A more accepted convention,

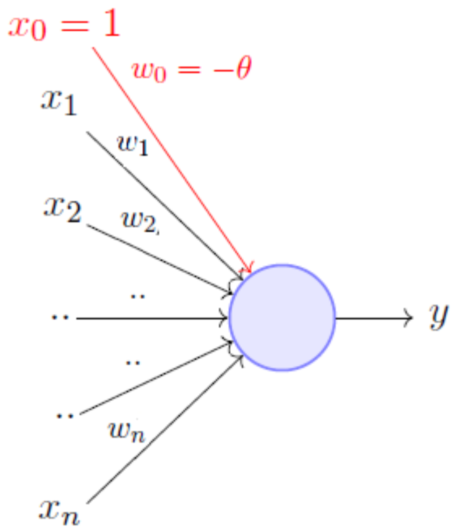
$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

Perceptron

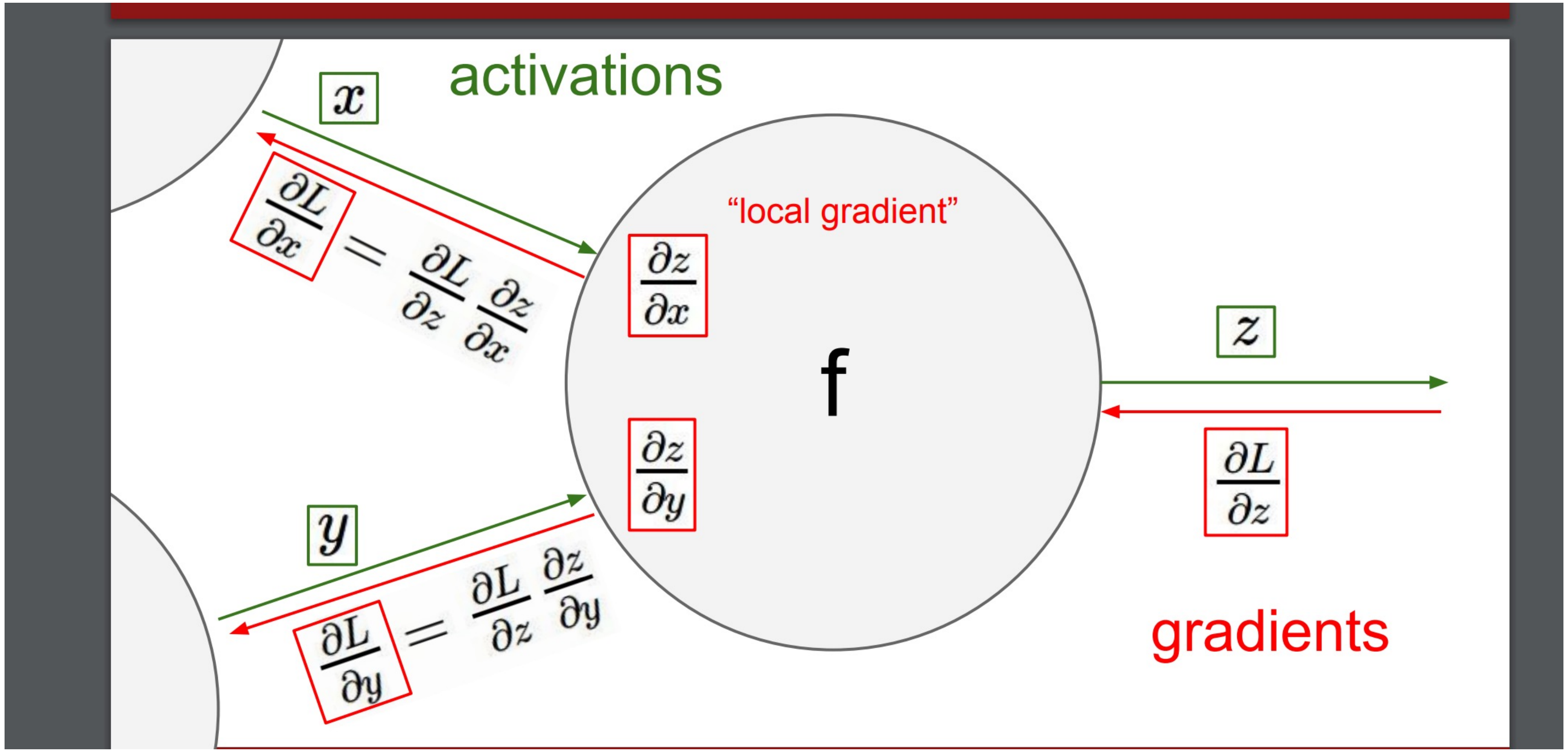
A single perceptron can only be used to implement **linearly separable** functions. It takes both real and boolean inputs and associates a set of **weights** to them, along with a **bias** (the threshold thing I mentioned above). We learn the weights, we get the function.



A more accepted convention,

$$
\begin{aligned}
y &= 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0 \\
&= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0
\end{aligned}
$$

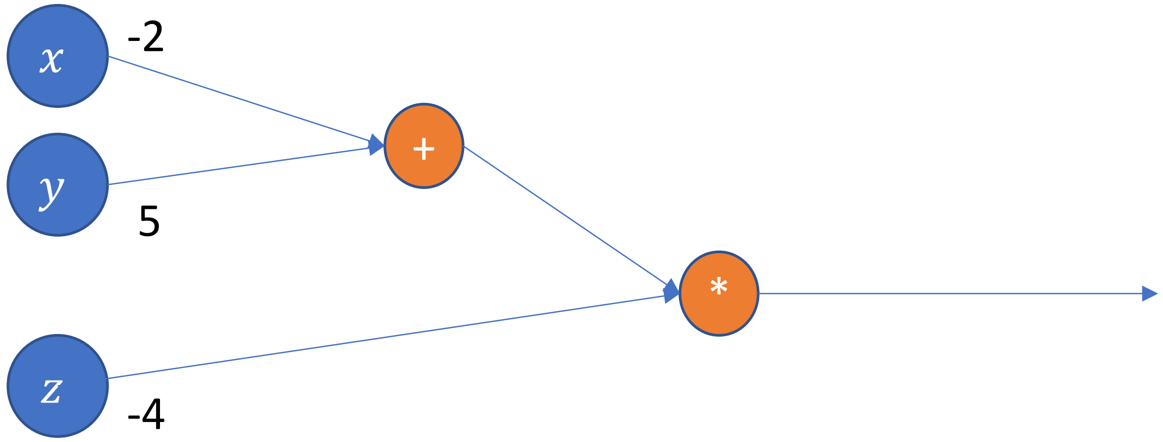
$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Single Neuron - Example

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

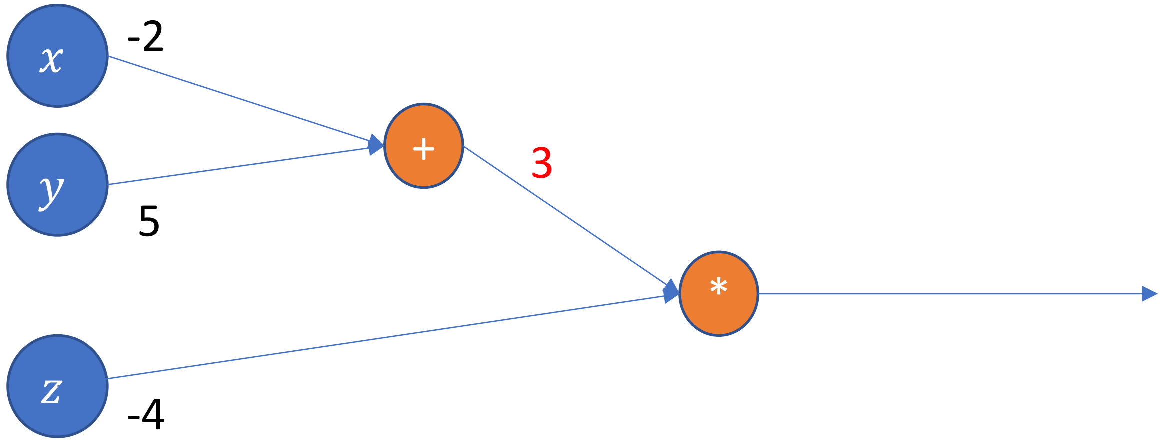
Example: $x = -2, y = 5, z = -4$



Single Neuron - Example

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

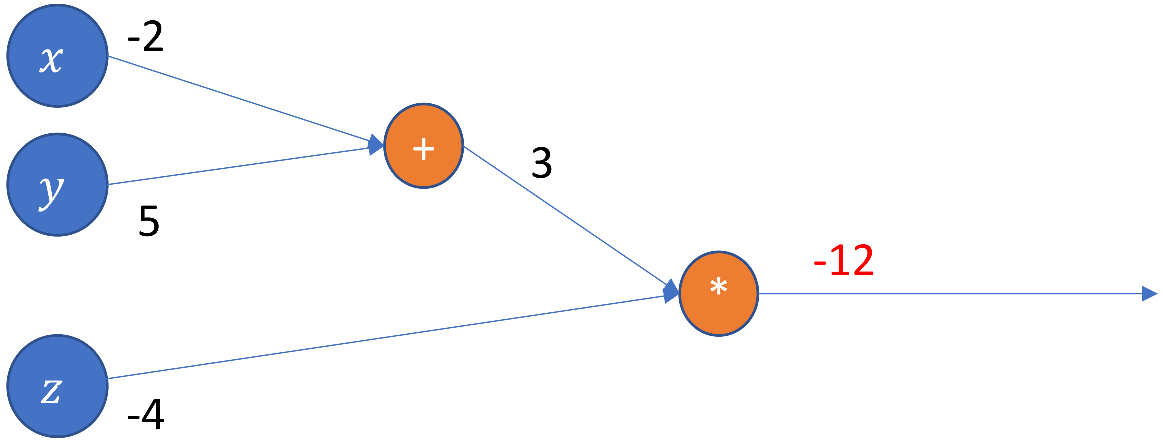
Example: $x = -2, y = 5, z = -4$



Single Neuron - Example

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

Example: $x = -2, y = 5, z = -4$



Single Neuron - Example

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

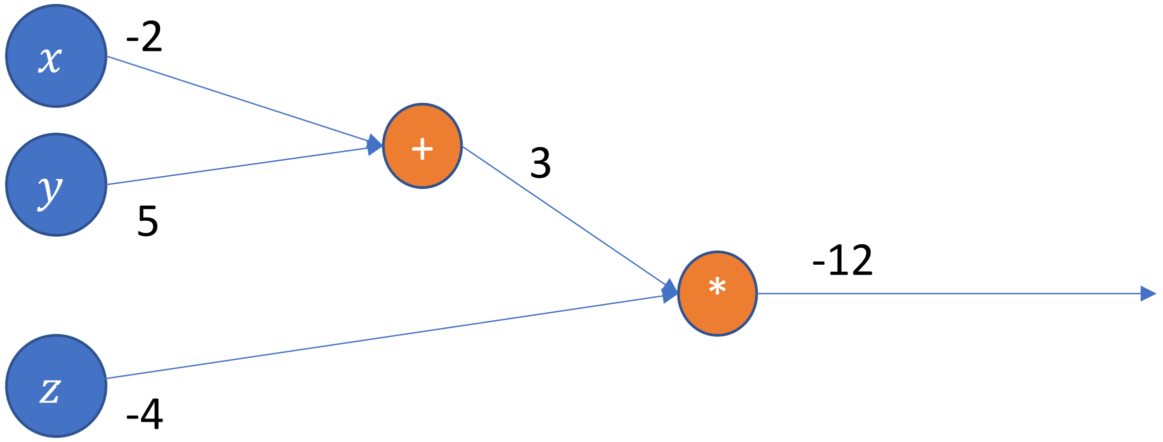
For the Gradient Descent –

$$q = x + y$$

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

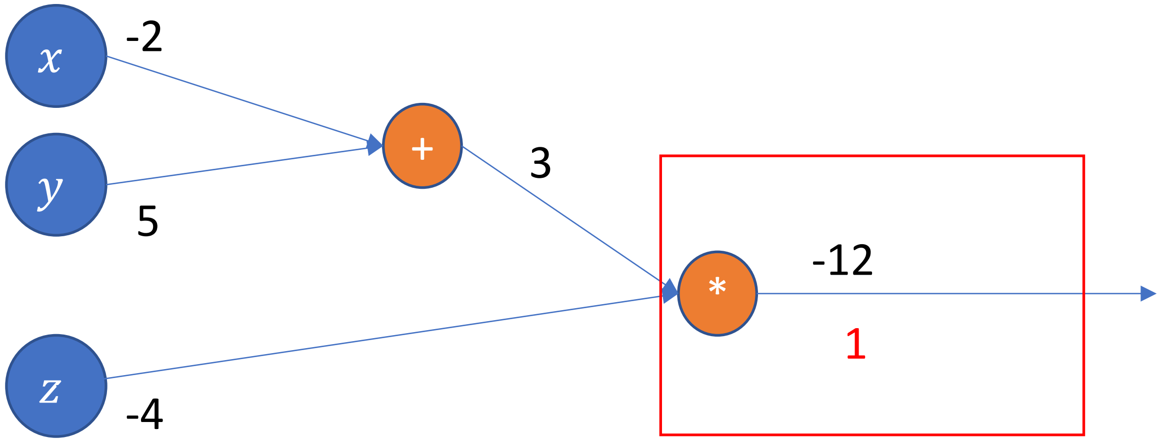
$$f = qz$$

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



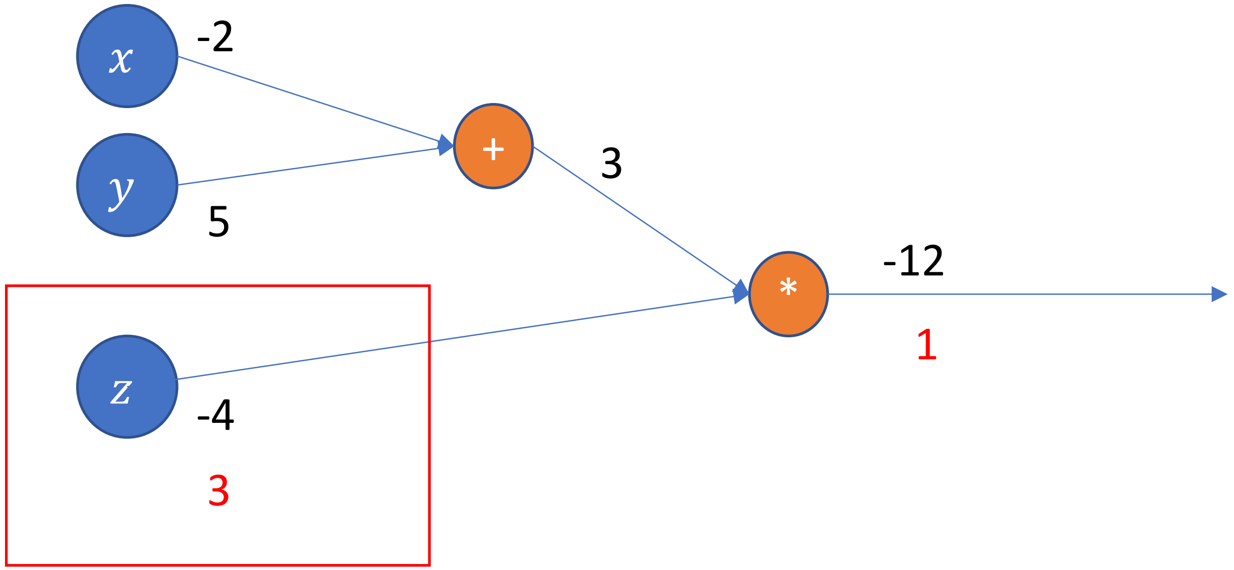
Single Neuron - Example

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



Single Neuron - Example

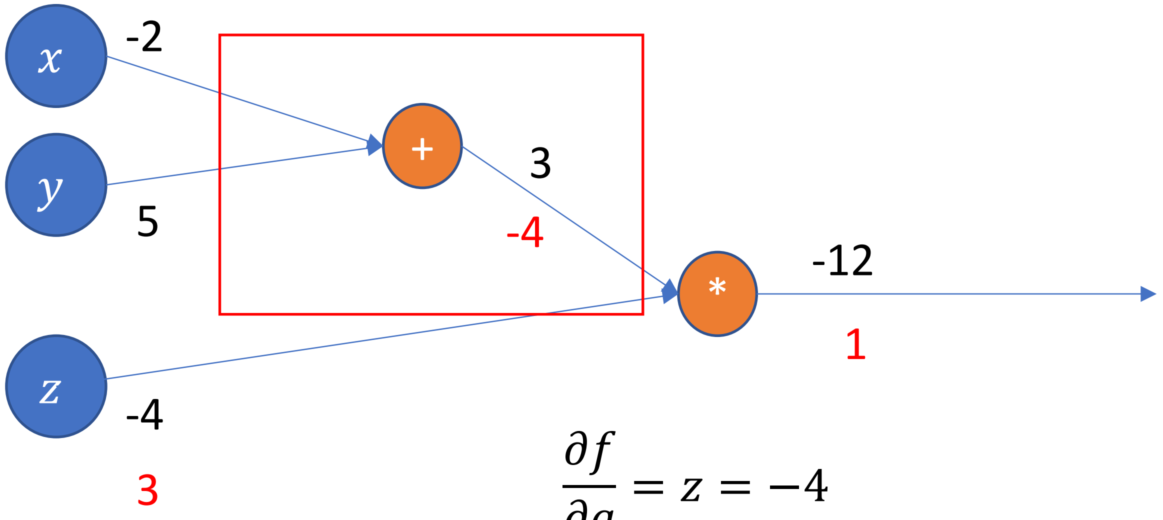
$$f(x, y, z) = (x + y)z$$



$$\frac{\partial f}{\partial z} = q = x + y = -2 + 5 = 3$$

Single Neuron - Example

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

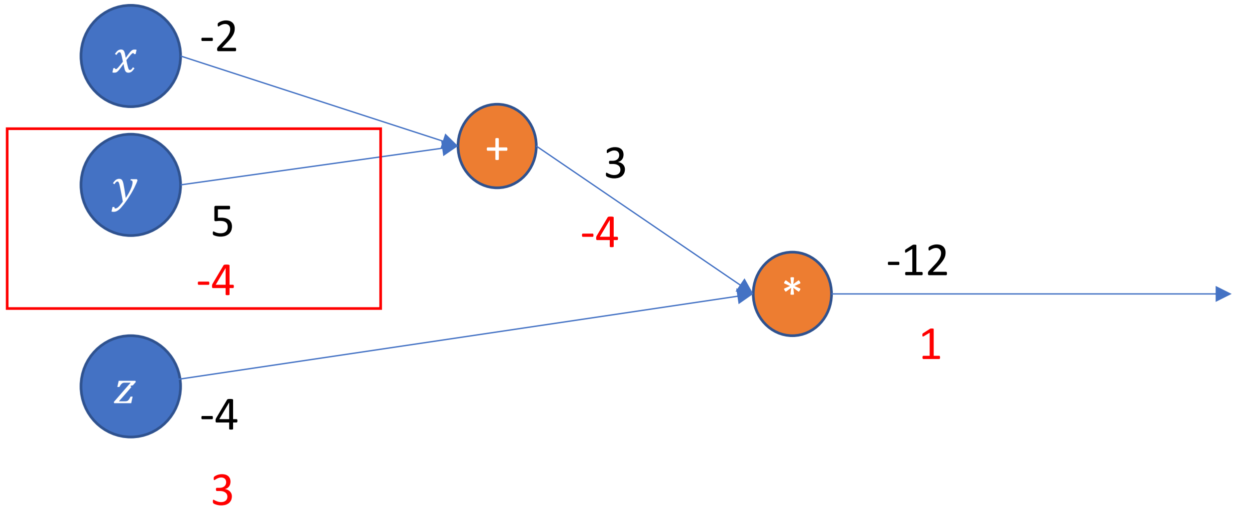


$$\frac{\partial f}{\partial q} = z = -4$$

Single Neuron - Example

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

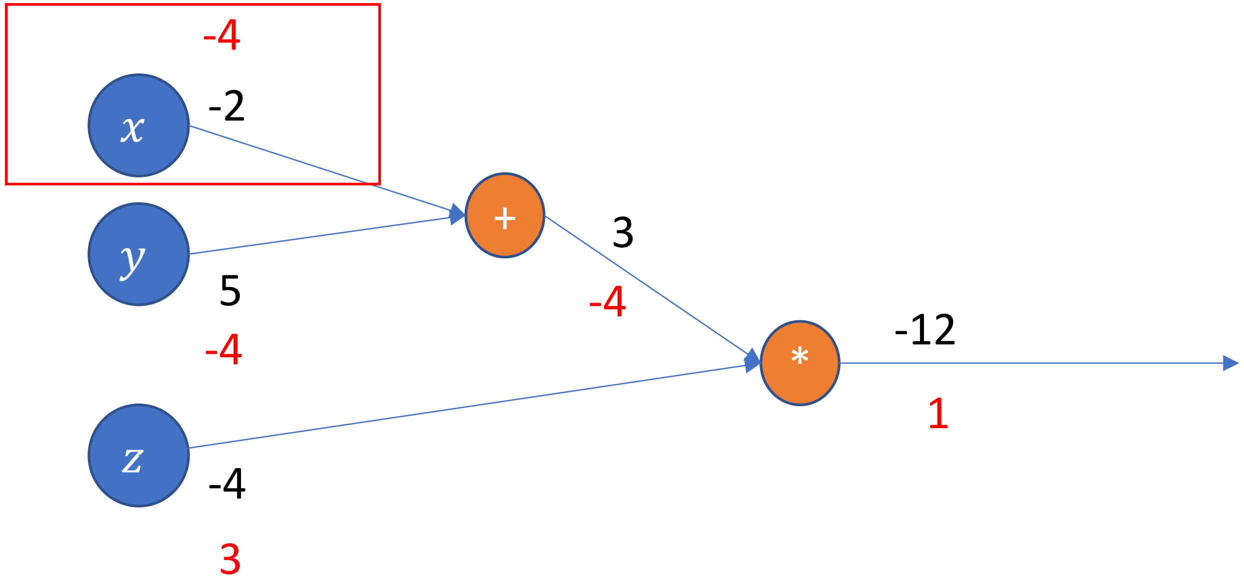
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = z \cdot 1 = -4$$



Single Neuron - Example

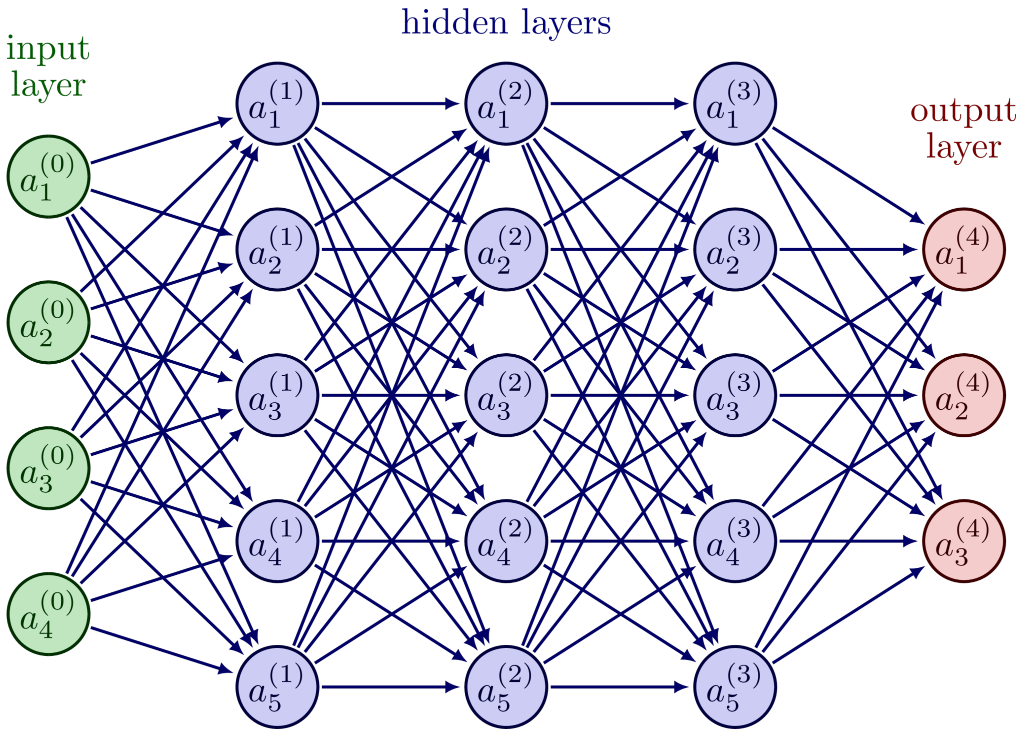
$$f(x, y, z) = (x + y)z$$

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



Multilayer Perceptron

Multilayer Perceptron also known as **Artificial Neural Networks** consists of more than one perception which is grouped together to form a multiple layer neural network.



Full Example

Neural Network - Example

Given the following learning function:

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

Given initialize values –

$$w_0 = 2, w_1 = -3, w_2 = -3$$

$$x_0 = -1, x_1 = -2$$

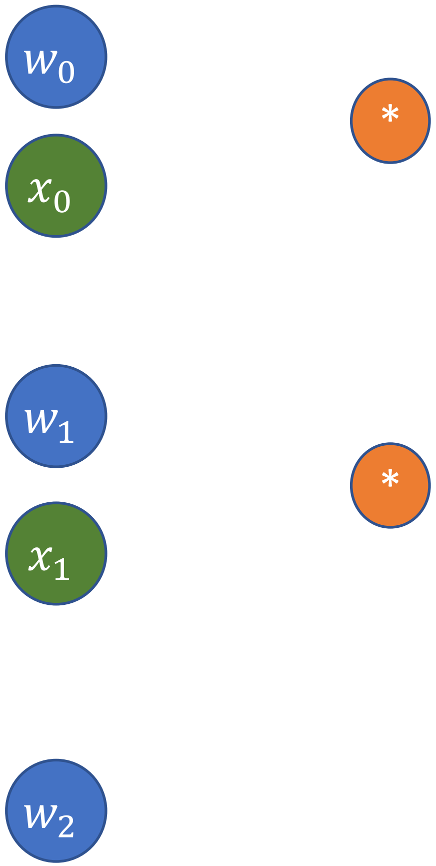
First, let's illustrate our network.

Neural Network - Example

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

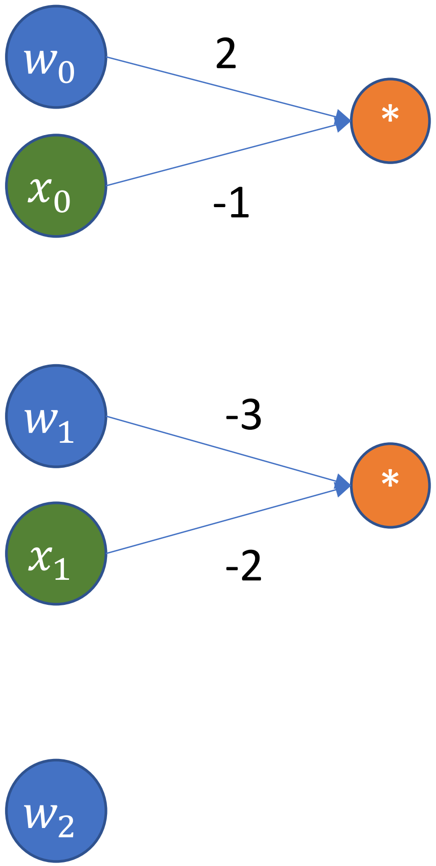
Neural Network - Example

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



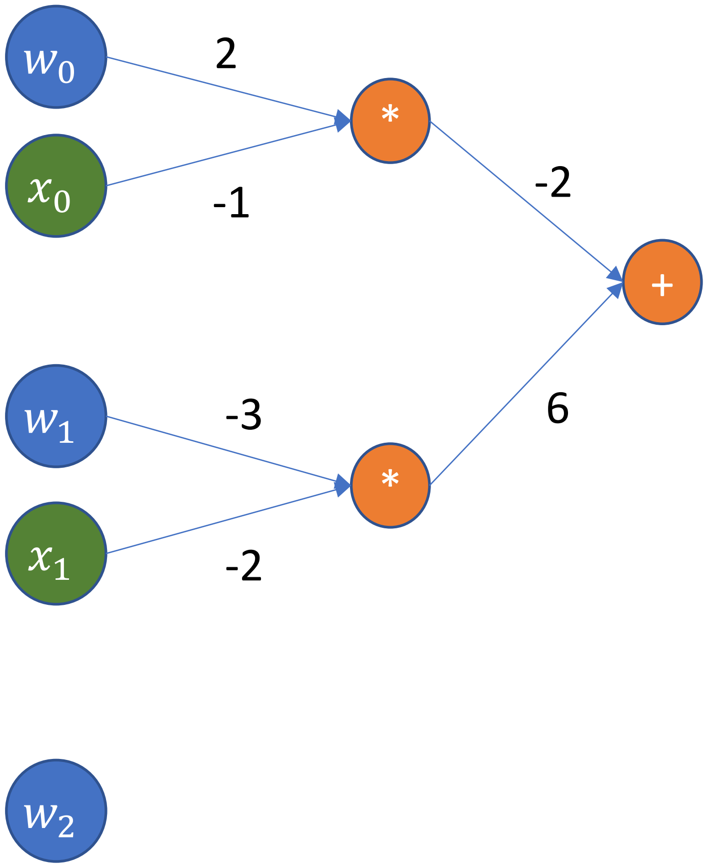
Neural Network - Example

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



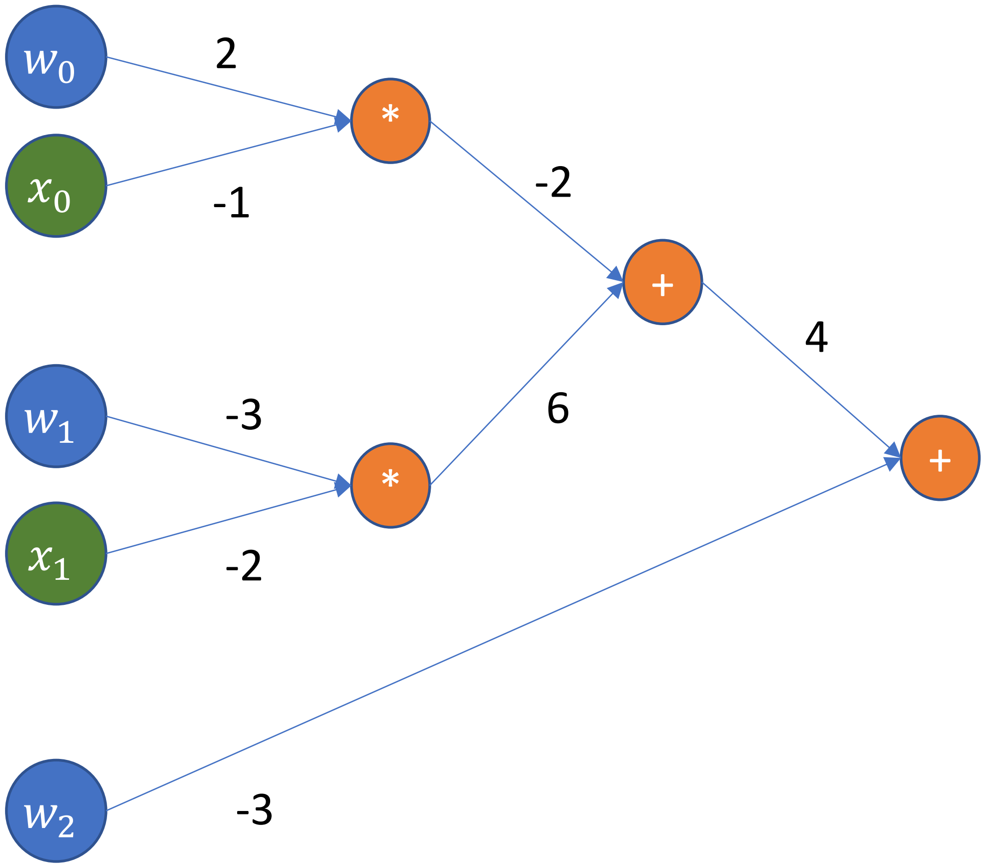
Neural Network - Example

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



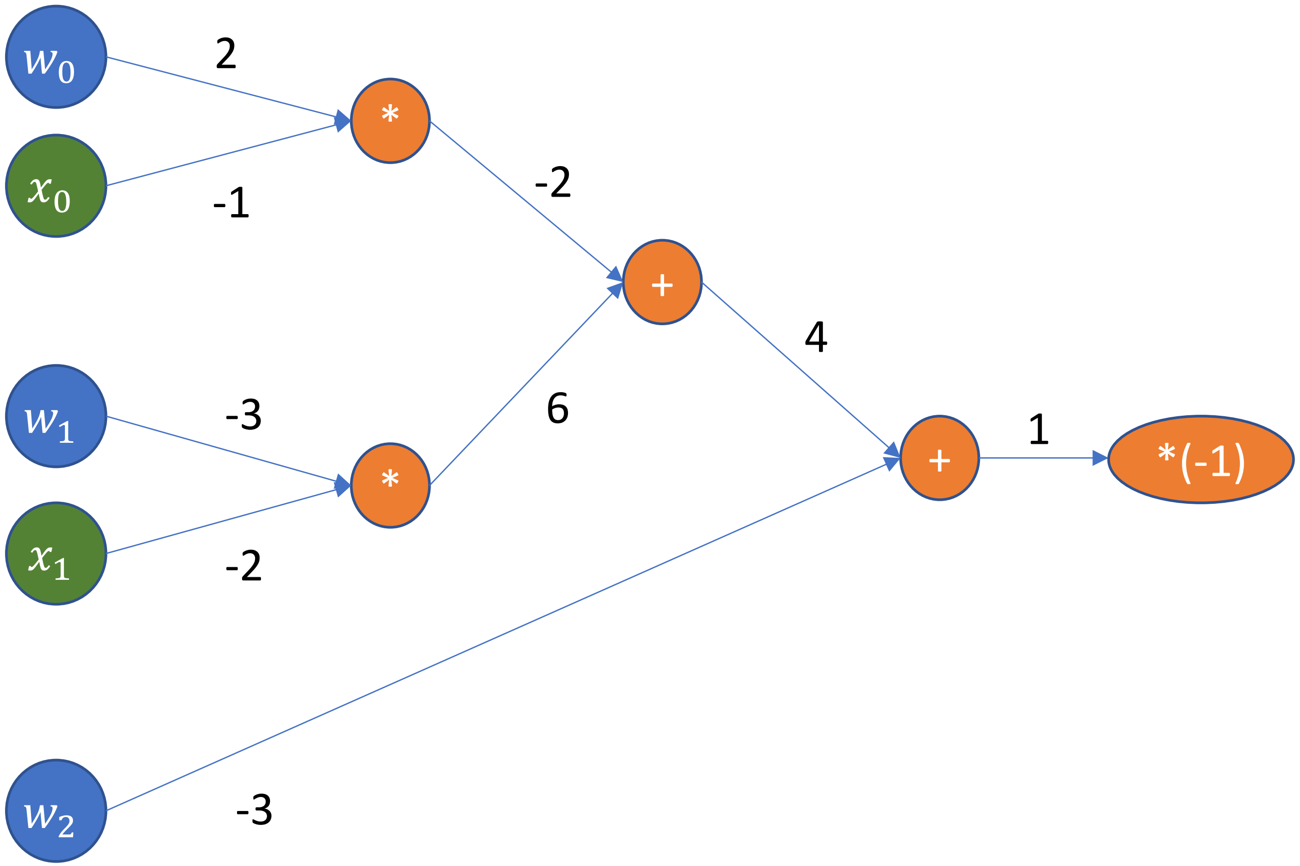
Neural Network - Example

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



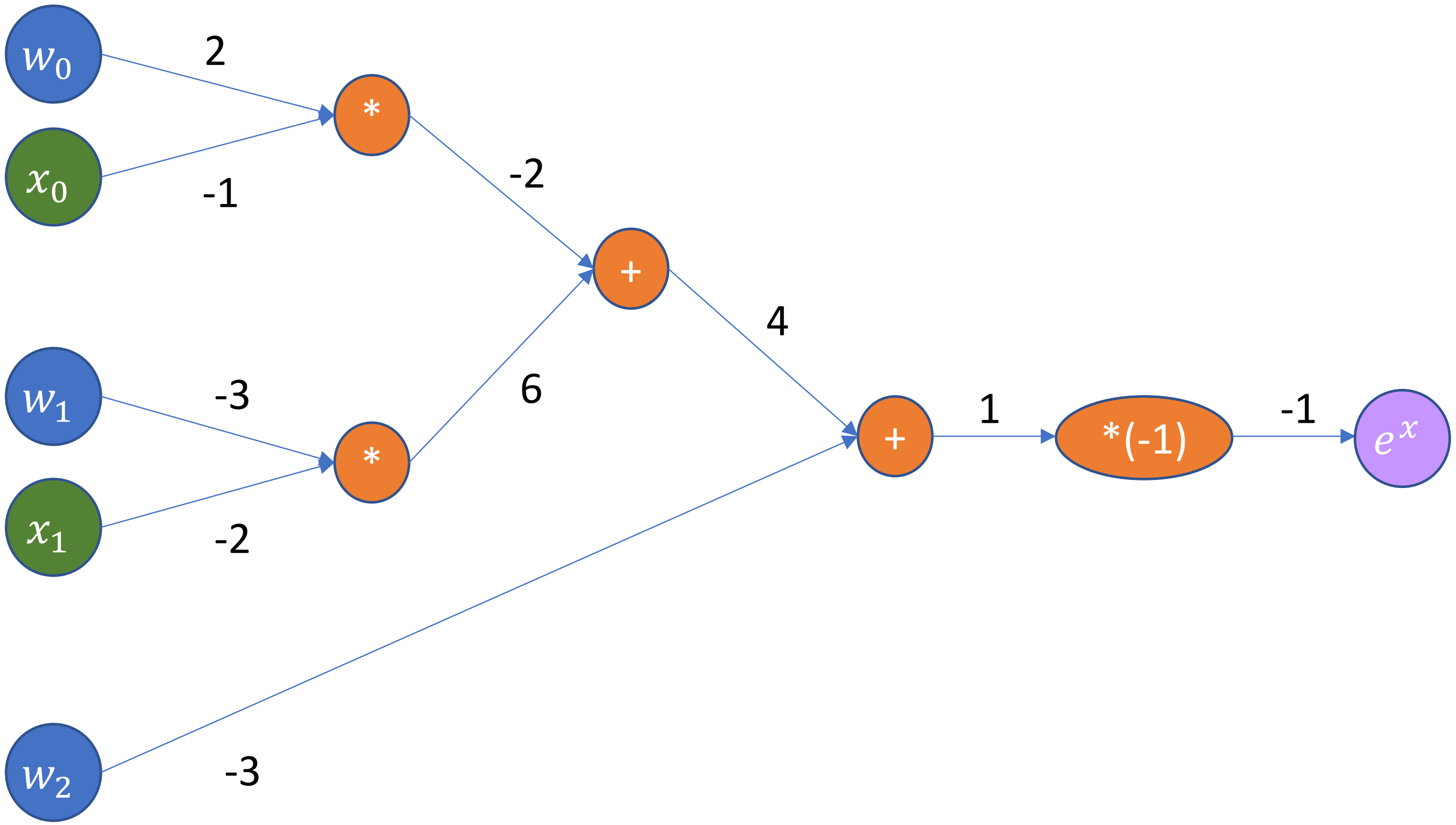
Neural Network - Example

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



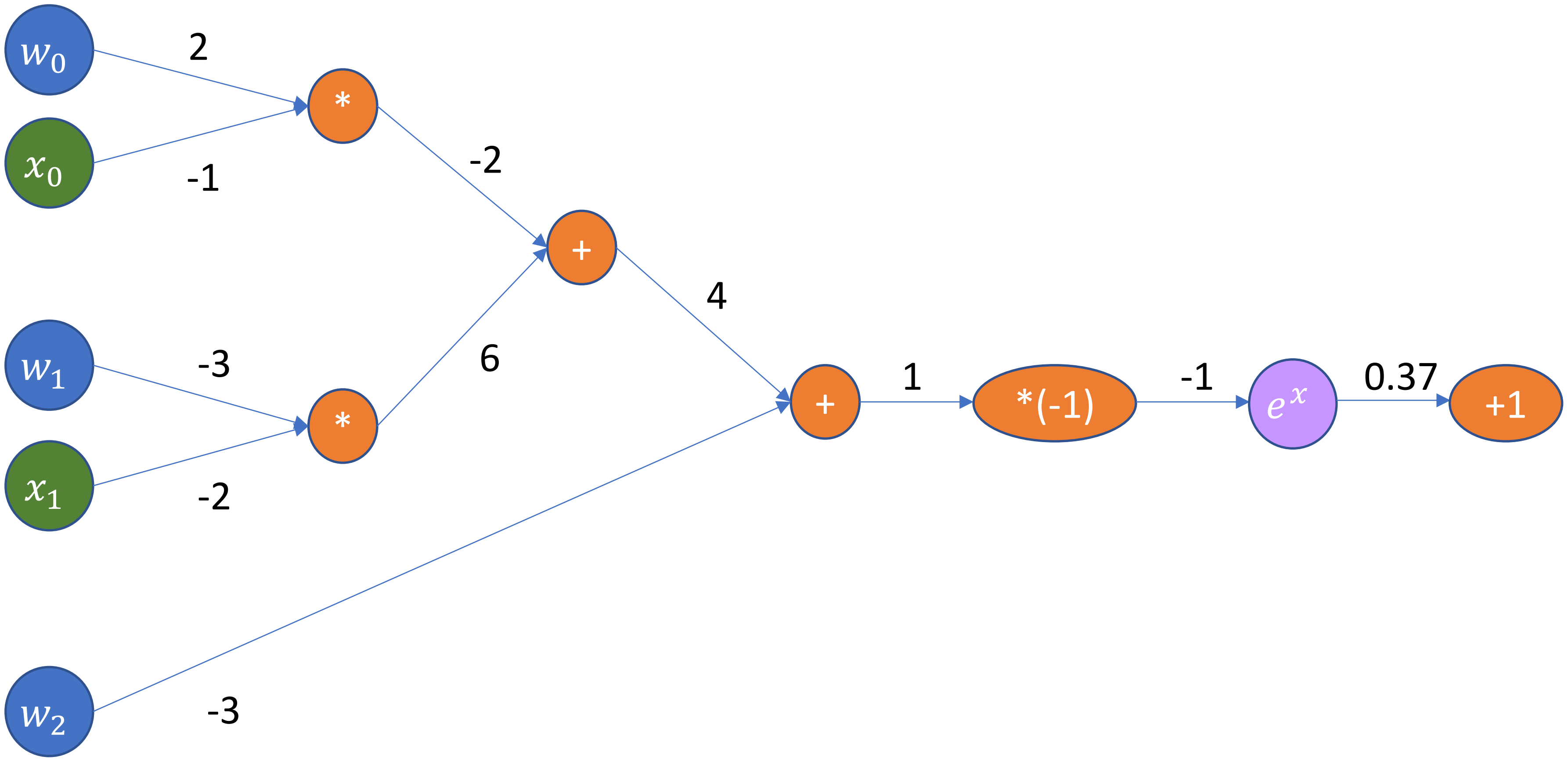
Neural Network - Example

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



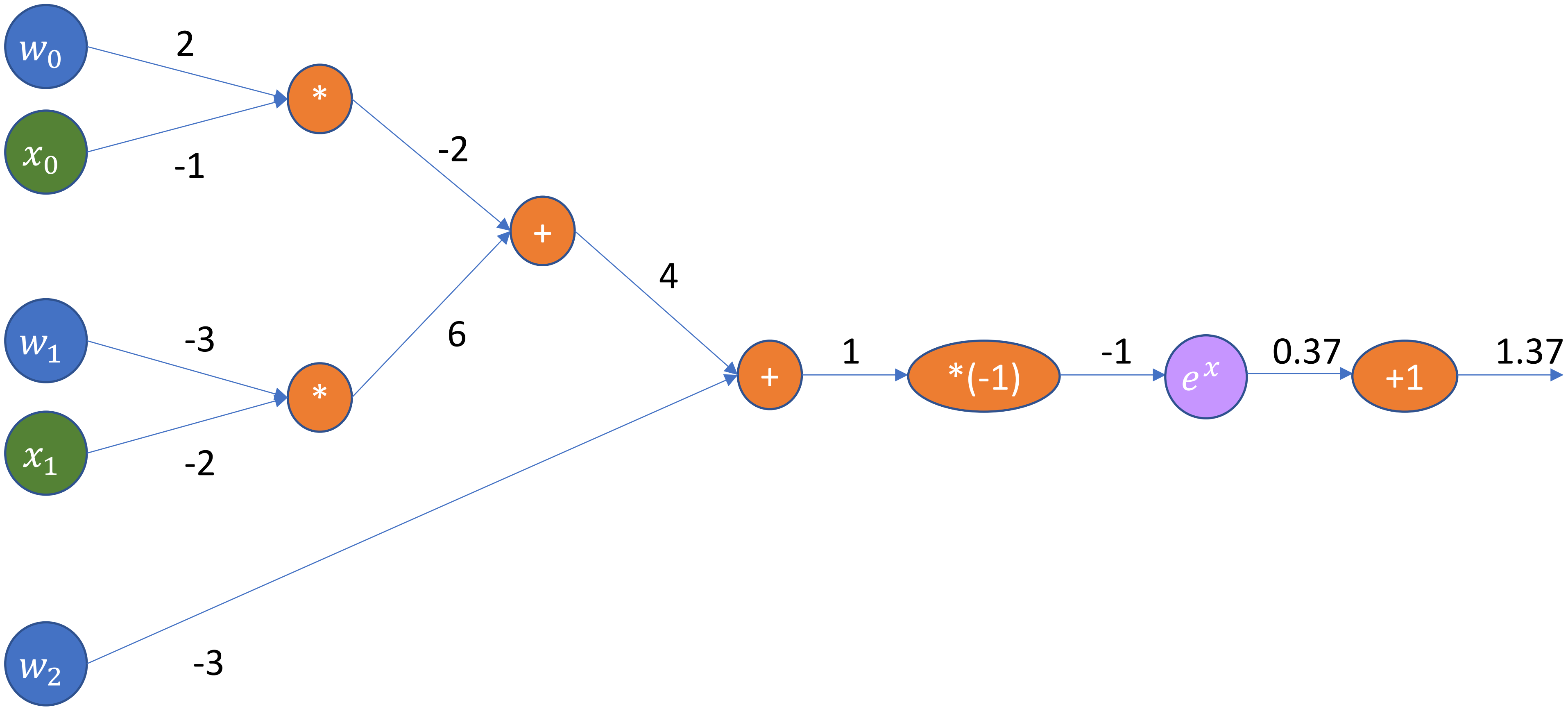
Neural Network - Example

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



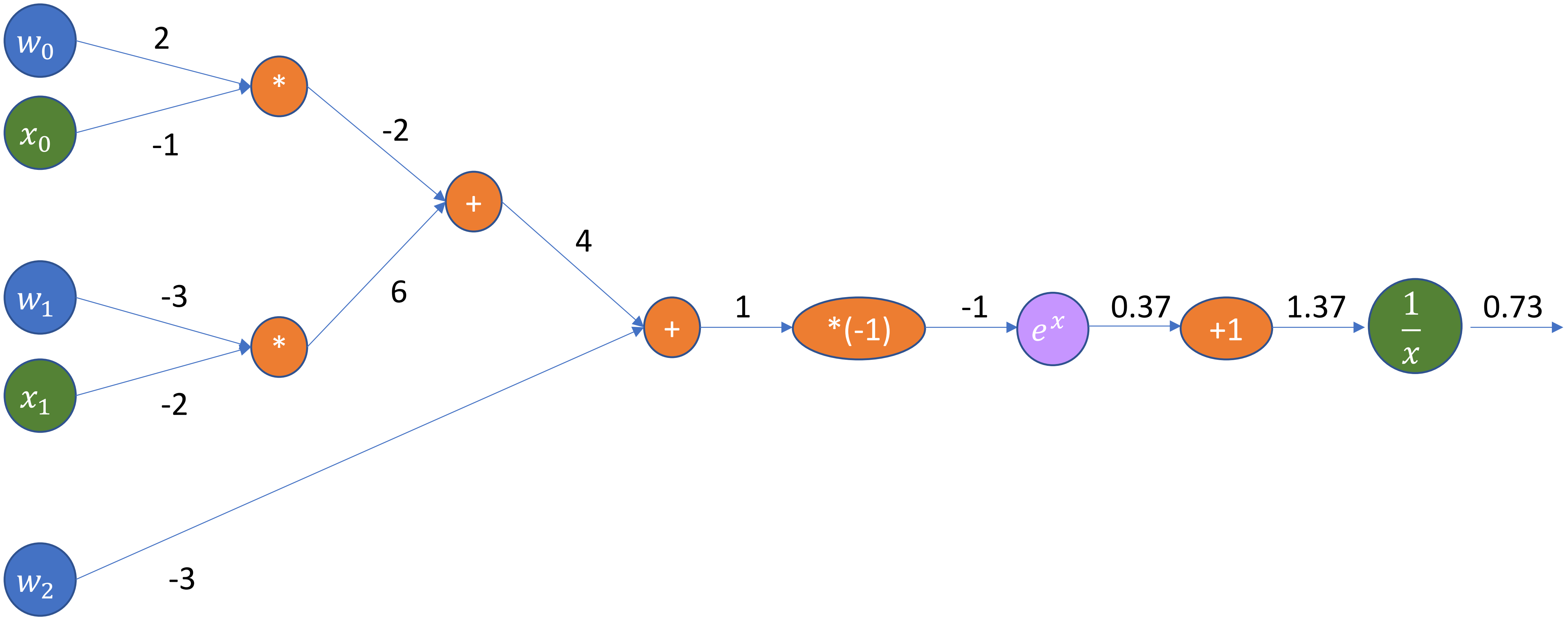
Neural Network - Example

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



Neural Network - Example

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



Neural Network - Example

For each layer in our network, define the function and its derivation

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

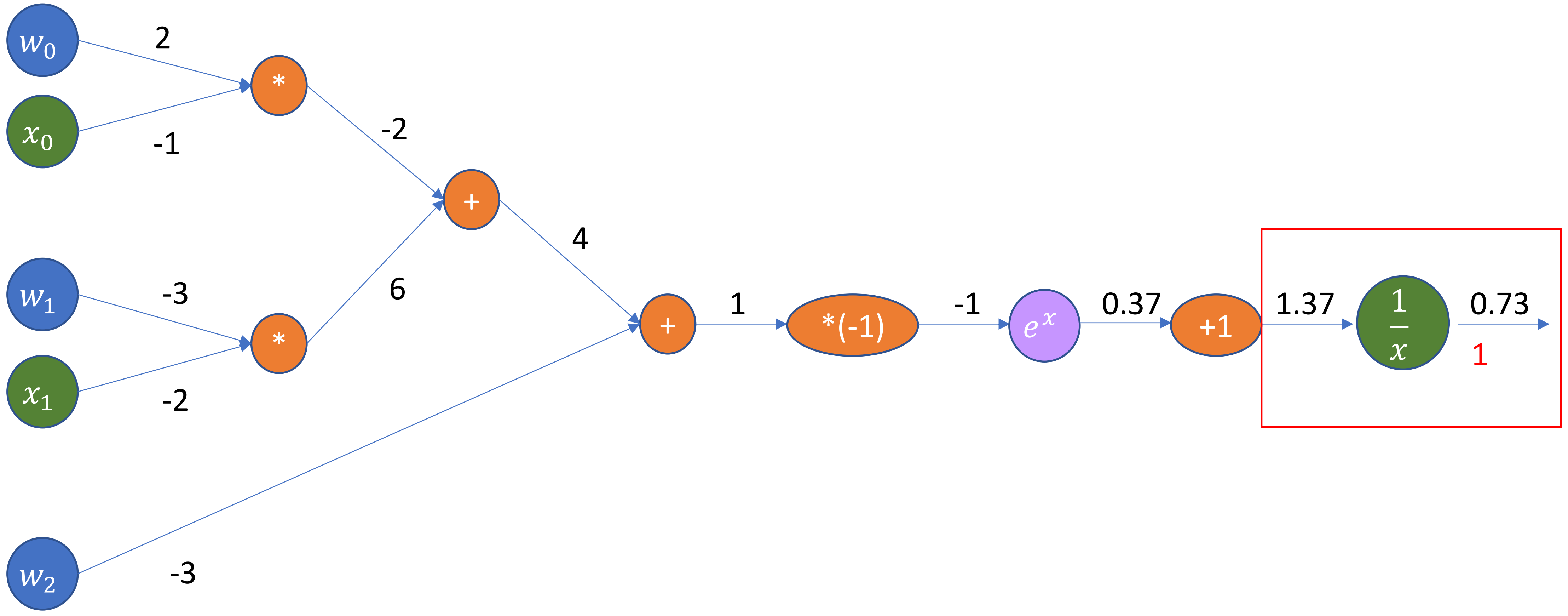
$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a$$

$$f(x) = x + c \rightarrow \frac{\partial f}{\partial x} = 1$$

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

Neural Network - Example

Start the back propagation updates

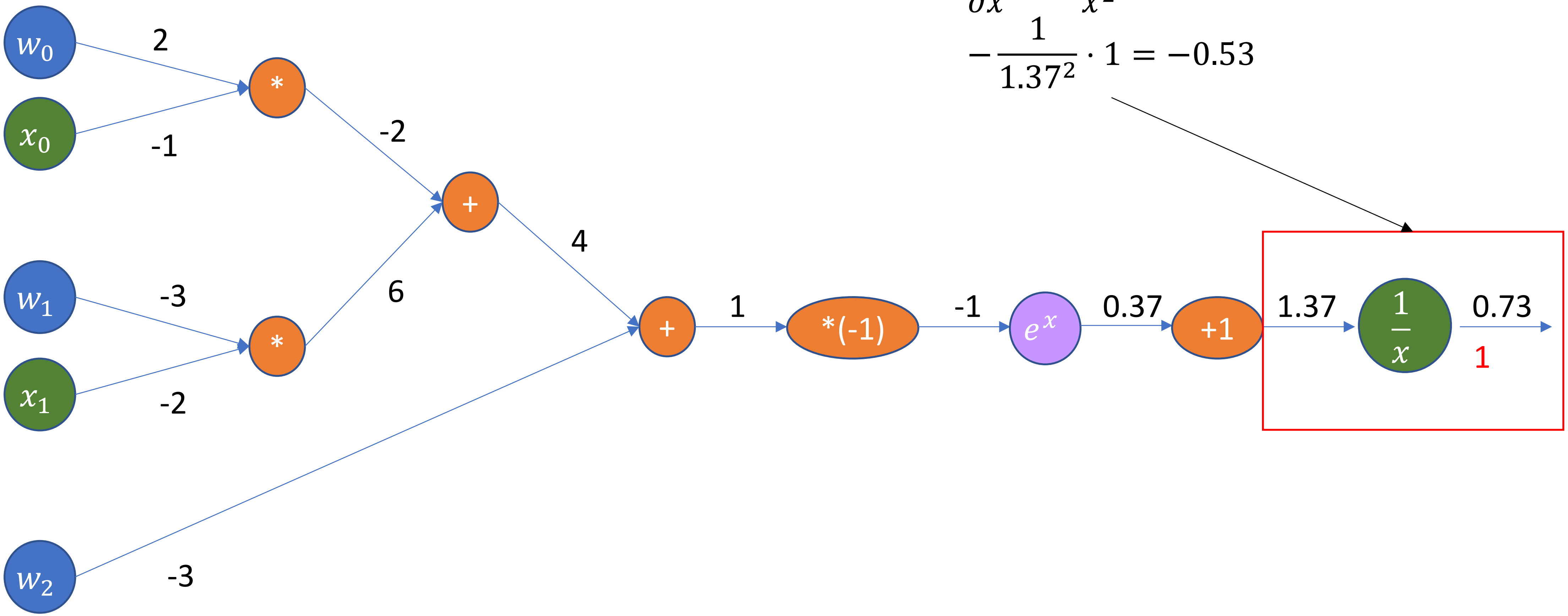


Neural Network - Example

Start the back propagation updates

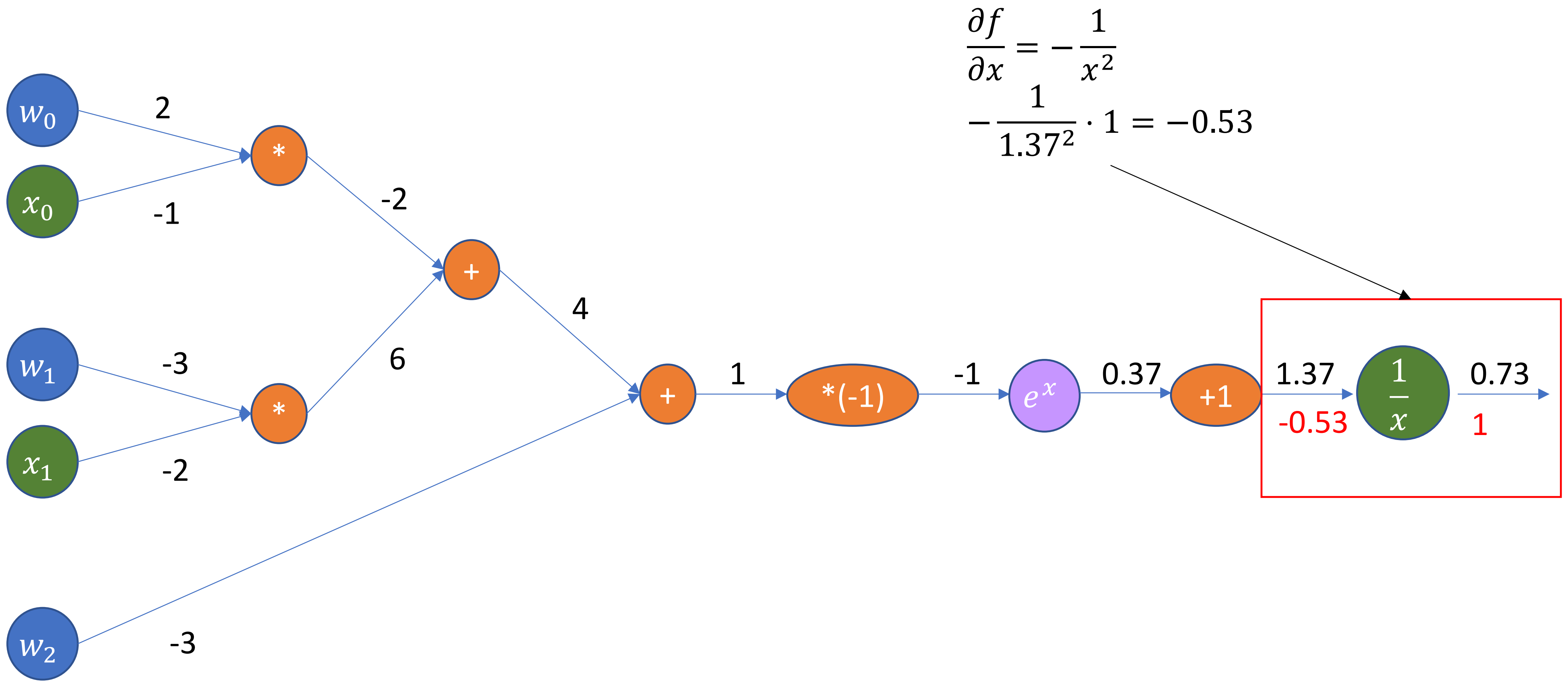
$$\frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$-\frac{1}{1.37^2} \cdot 1 = -0.53$$



Neural Network - Example

Start the back propagation updates

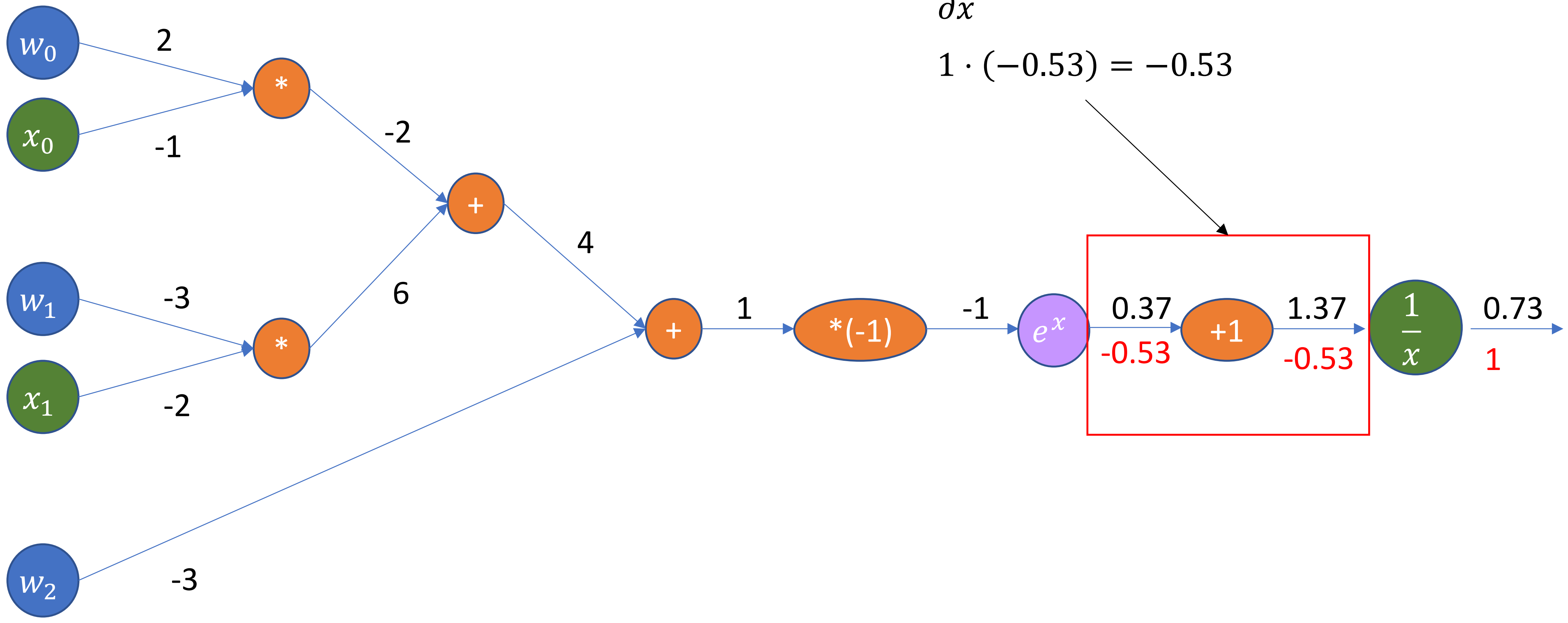


Neural Network - Example

Start the back propagation updates

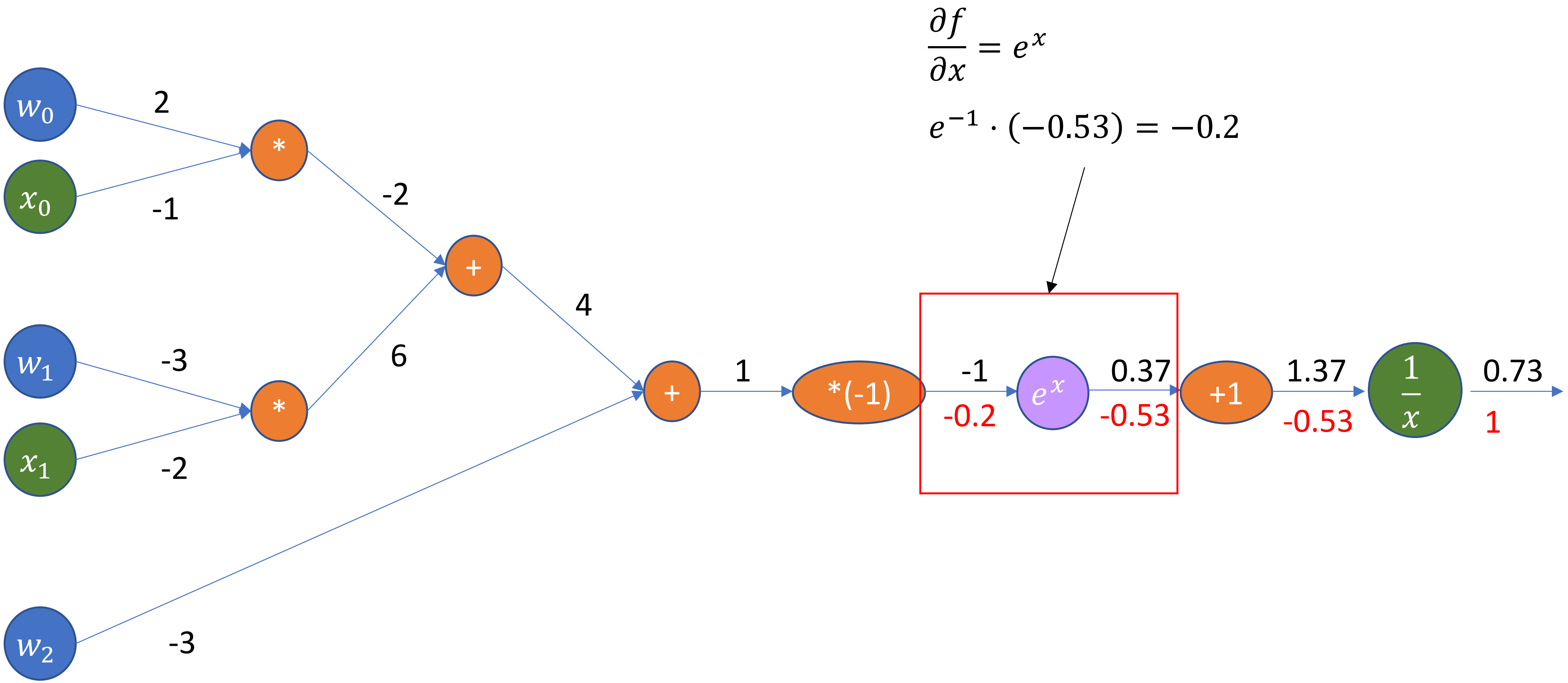
$$\frac{\partial f}{\partial x} = 1$$

$$1 \cdot (-0.53) = -0.53$$



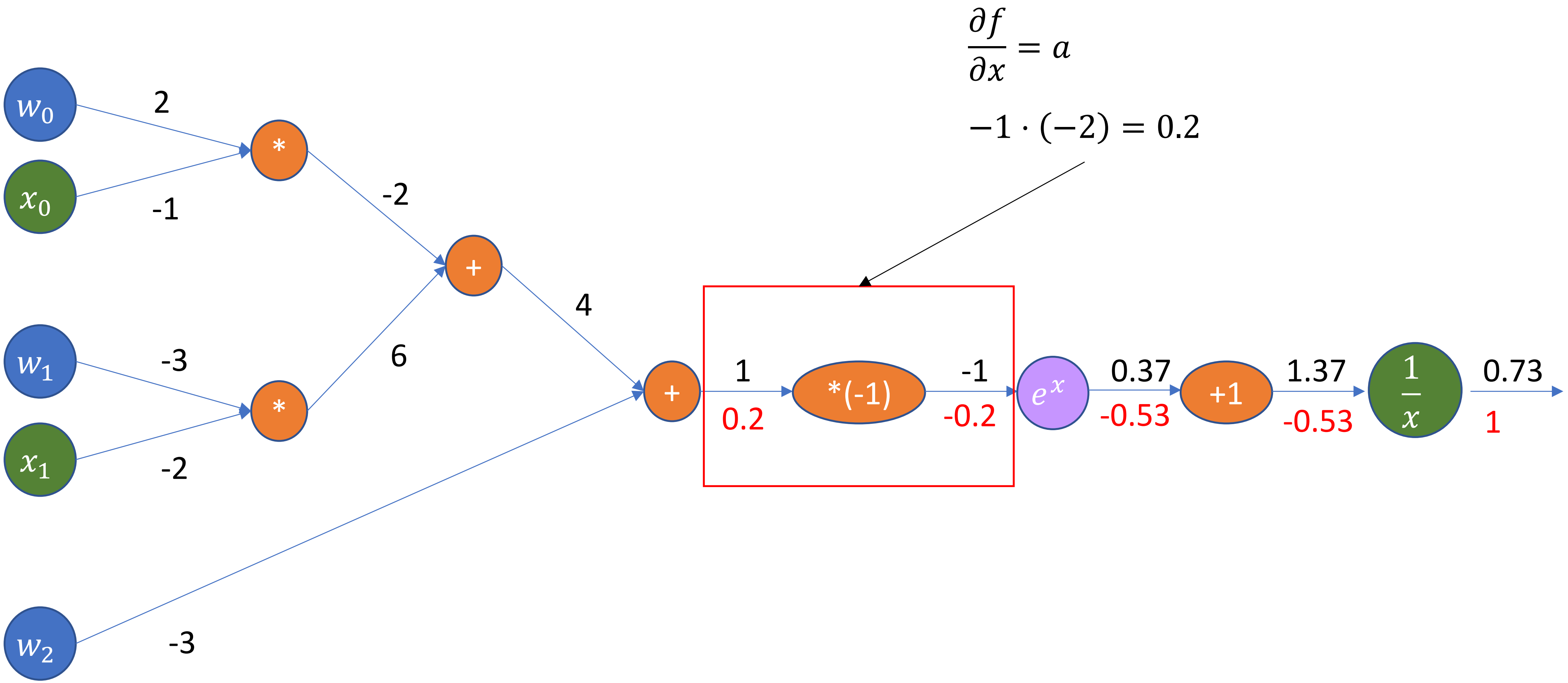
Neural Network - Example

Start the back propagation updates



Neural Network - Example

Start the back propagation updates



Neural Network - Example

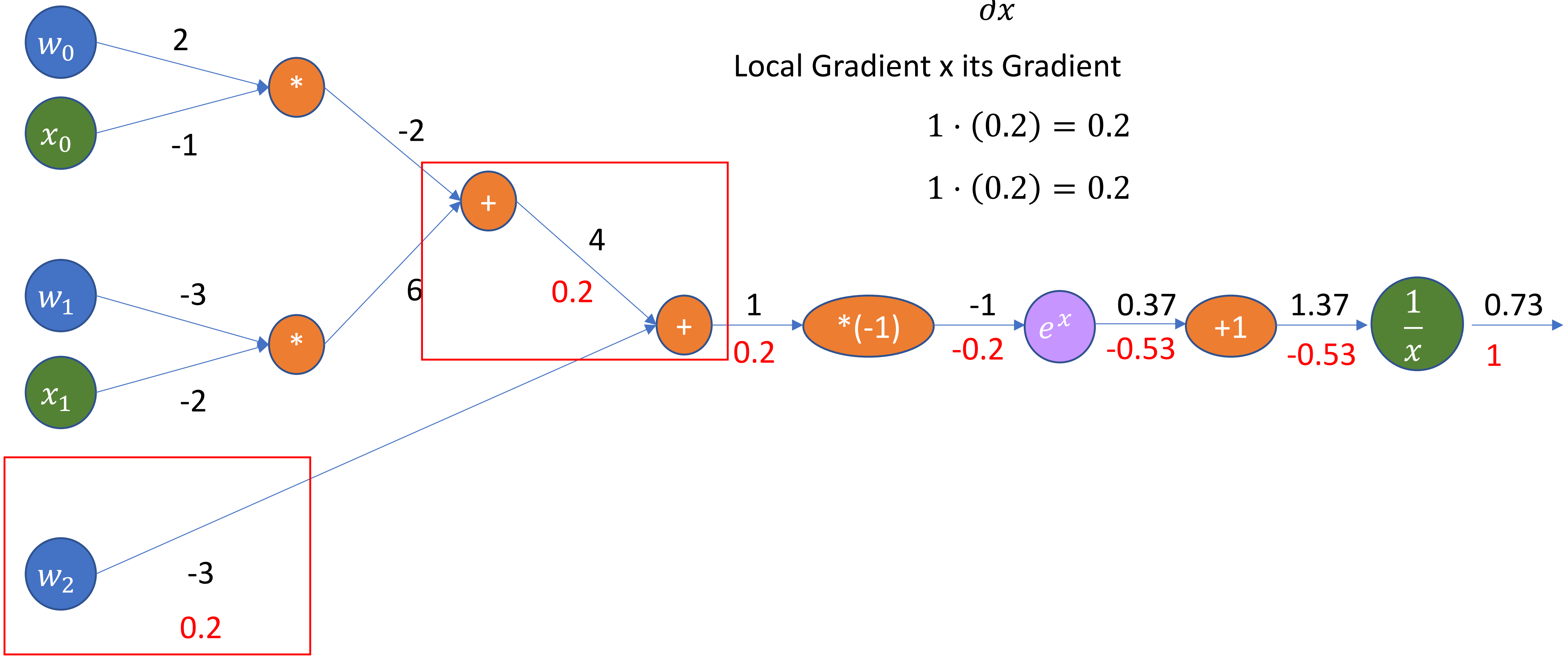
Start the back propagation updates

$$\frac{\partial f}{\partial x} = 1$$

Local Gradient x its Gradient

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

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

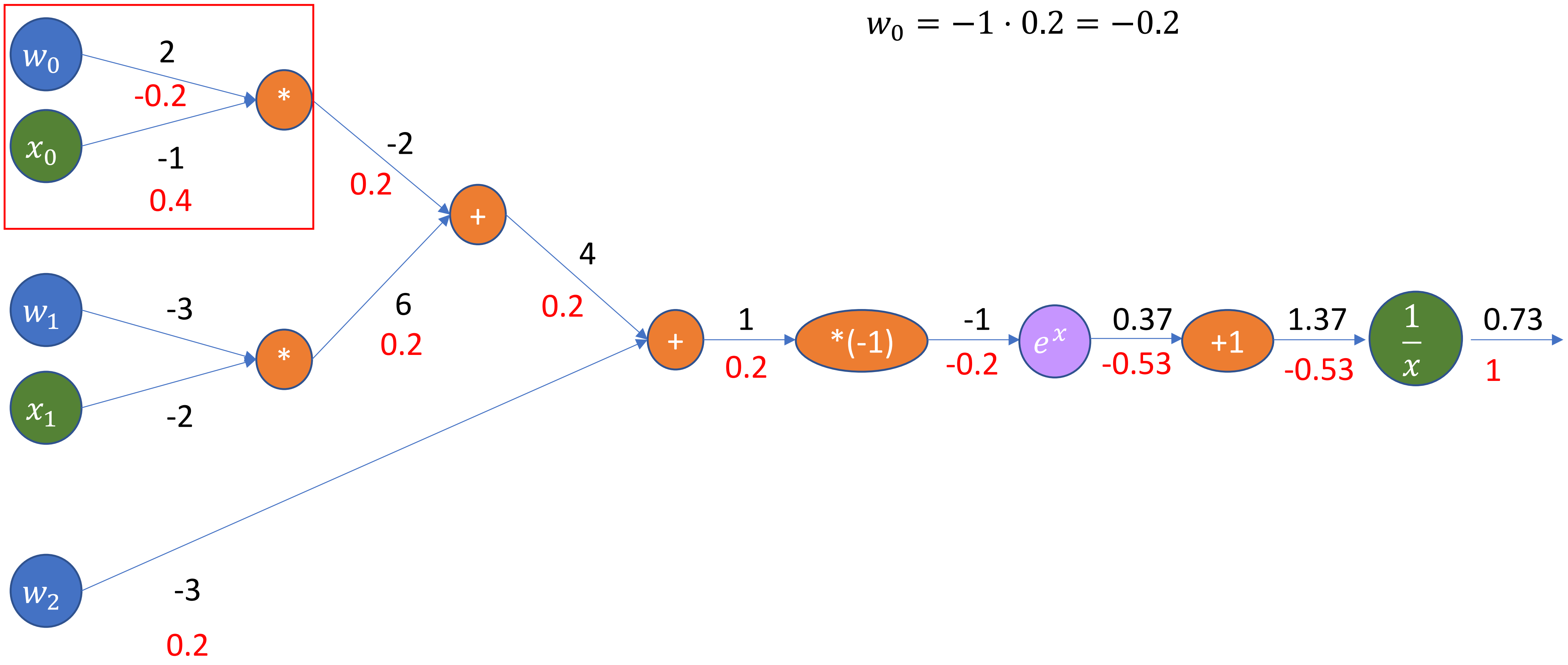


Neural Network - Example

Start the back propagation updates

$$x_0 = 2 \cdot 0.2 = 0.4$$

$$w_0 = -1 \cdot 0.2 = -0.2$$



Neural Network - Example

Start the back propagation updates

$$x_1 = -3 \cdot 0.2 = -0.6$$

$$w_1 = -2 \cdot 0.2 = -0.4$$