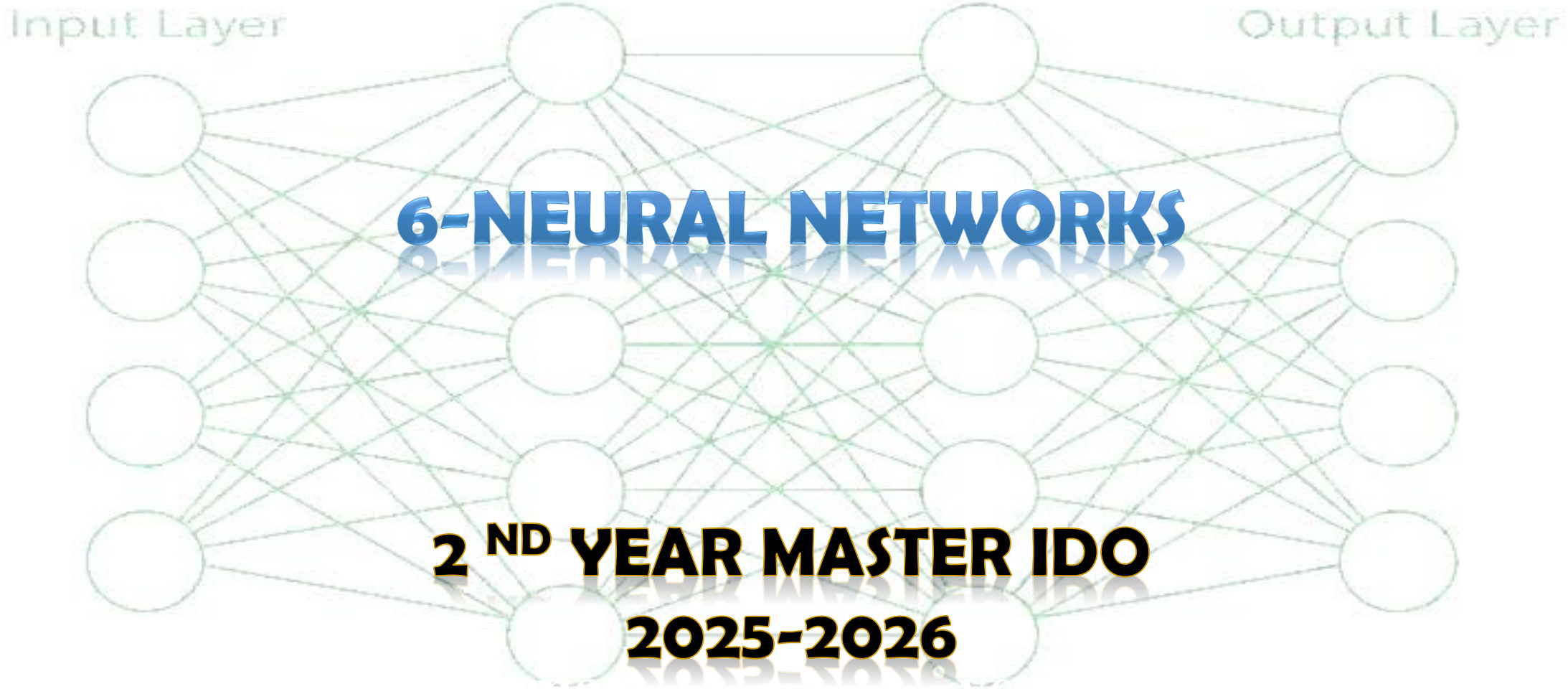
M'sila University
Faculty of Mathematics and computer science
Department of computer science



6-NEURAL NETWORKS

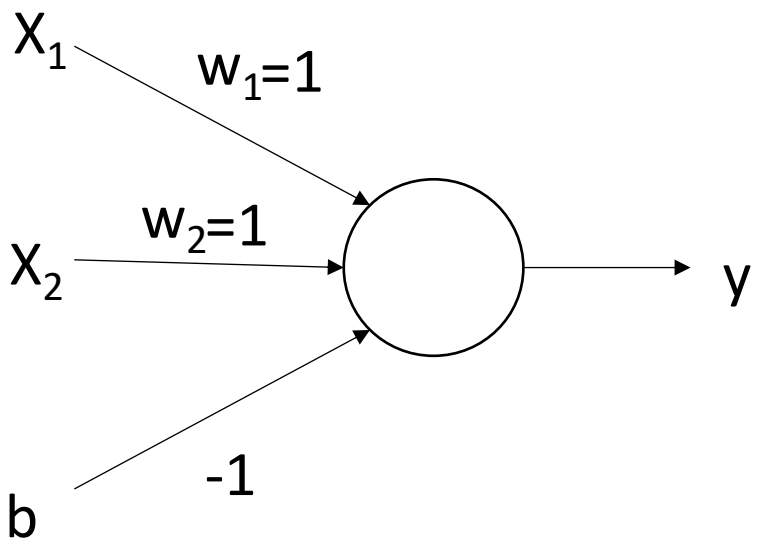
2 ND YEAR MASTER IDO
2025-2026

In the beginning of neural networks there was something called the perceptron :

The **perceptron** is one of the simplest types of artificial neural networks, designed for binary classification tasks. It was introduced by **Frank Rosenblatt** in 1958 and represents an early foundational model in machine learning.

Perceptron for the function AND

$$y = \begin{cases} 0 \; if \; w.x + b \; \leq 0 \\ 1 \; if \; w.x + b \; > 0 \end{cases}$$

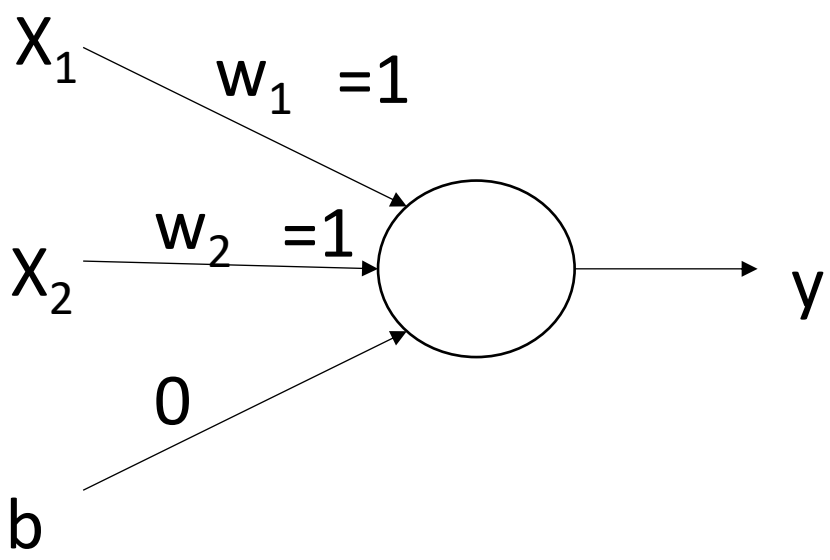


AND		
X1	X2	y
0	0	0
1	0	0
0	1	0
1	1	1

Perceptron for the function OR

$$y = \begin{cases} 0 \; if \; w.x + b \; \leq 0 \\ 1 \; if \; w.x + b \; > 0 \end{cases}$$

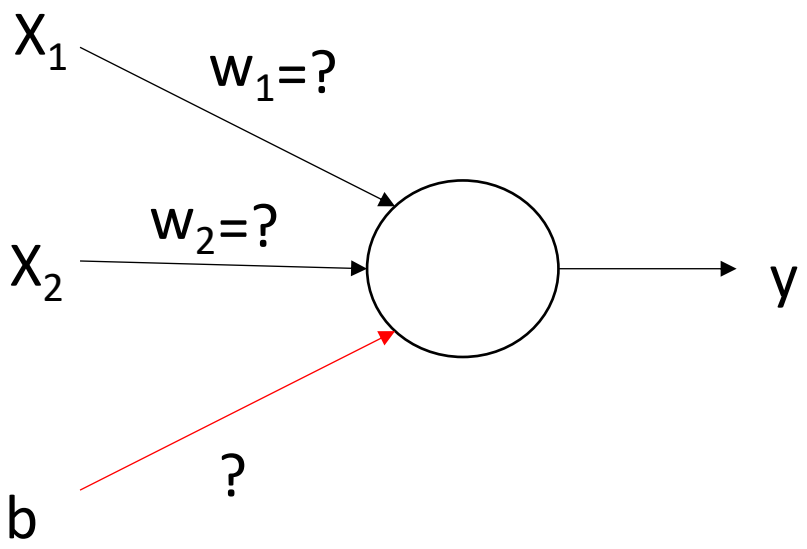
OR		
X1	X2	y
0	0	0
1	0	1
0	1	1
1	1	1



Perceptron for the function XOR

$$y = \begin{cases} 0 \; if \; w.x + b \; \leq 0 \\ 1 \; if \; w.x + b \; > 0 \end{cases}$$

XOR		
X1	X2	y
0	0	0
1	0	1
0	1	1
1	1	0



The equation of the perceptron is a linear equation.

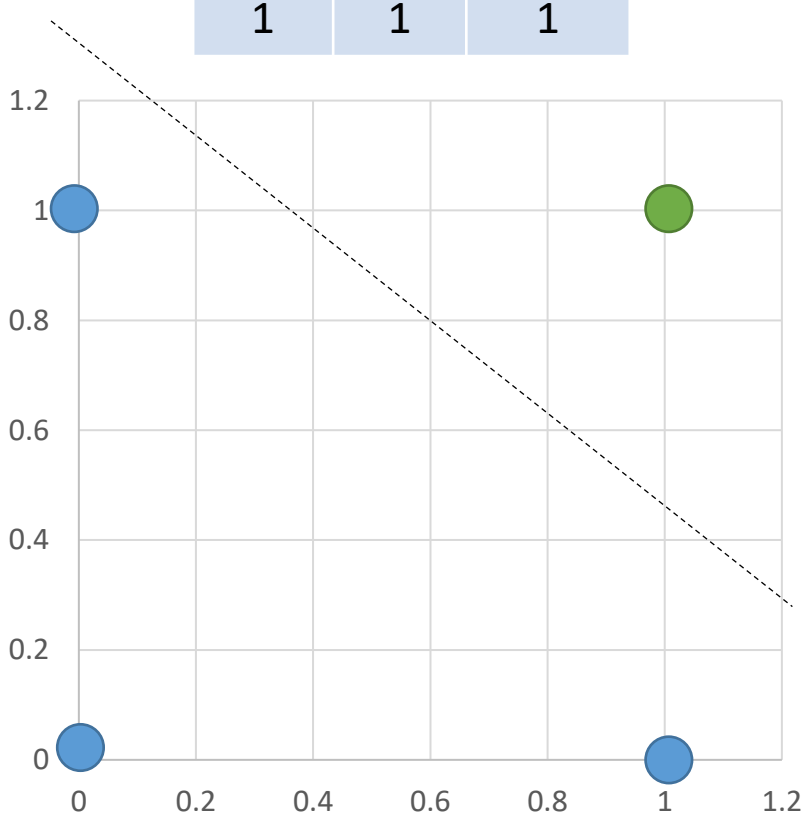
$$w_1 x_1 + w_2 x_2 + b = 0$$

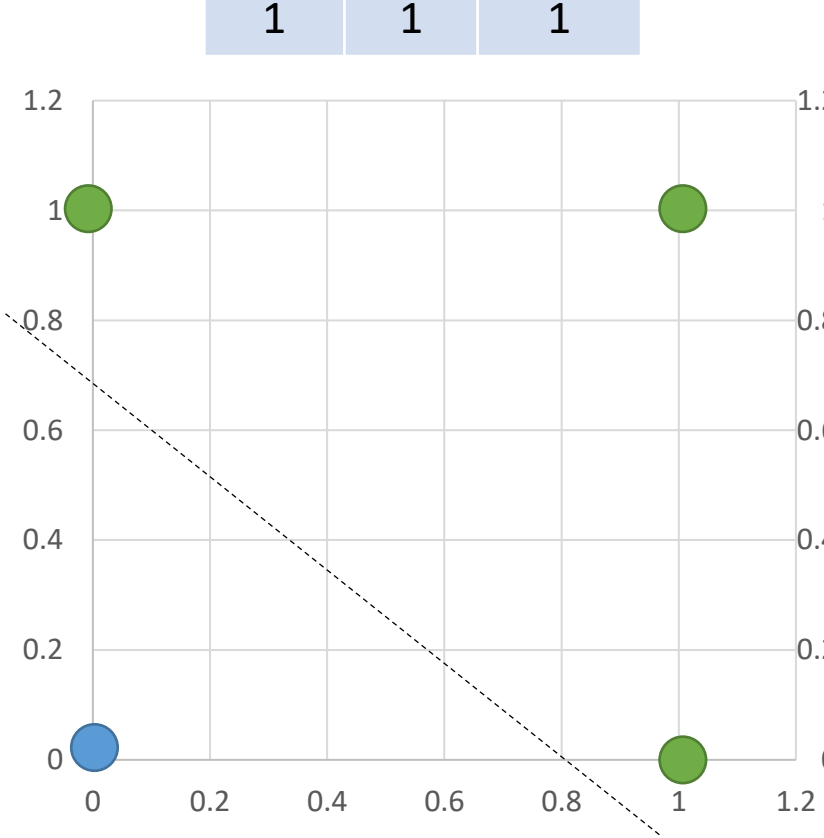
$$x_2 = -(w_1/w_2)\, x_1 - b/w_2$$

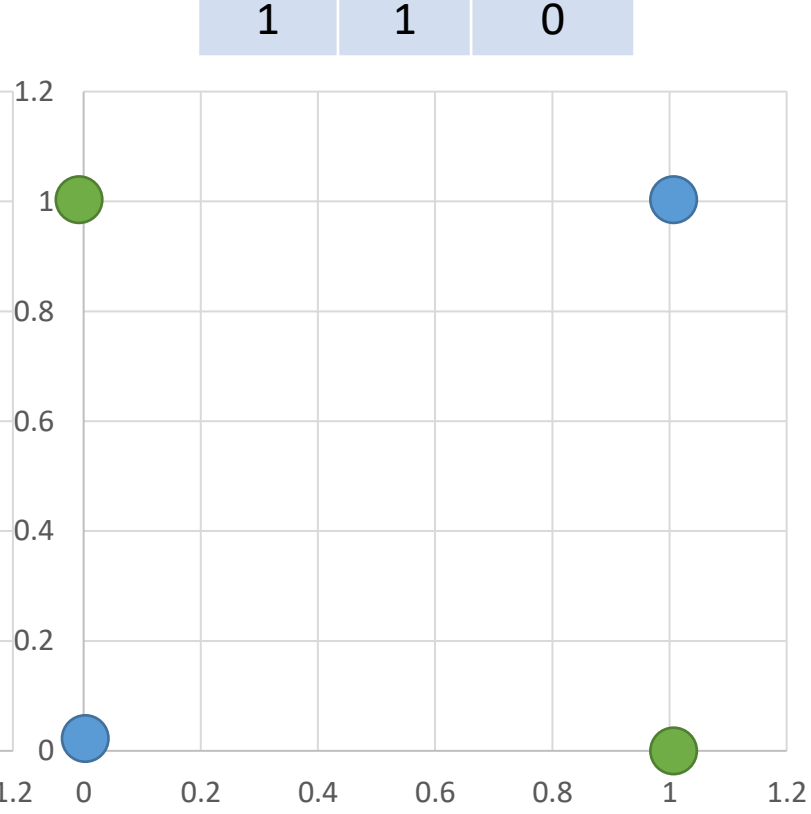
AND		
X1	X2	y
0	0	0
1	0	0
0	1	0
1	1	1

OR		
X1	X2	y
0	0	0
1	0	1
0	1	1
1	1	1

XOR		
X1	X2	y
0	0	0
1	0	1
0	1	1
1	1	0



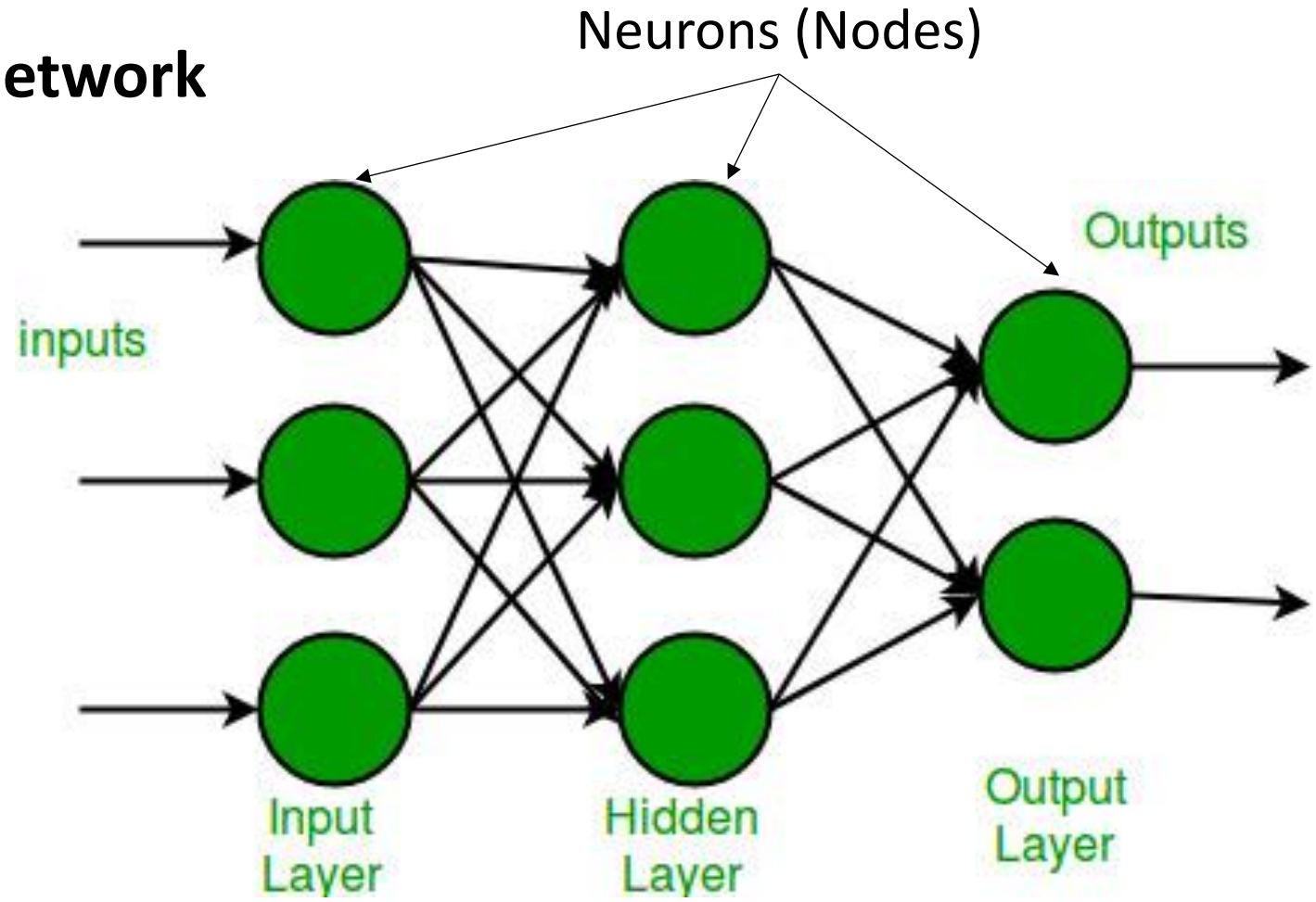




XOR is not a linearly separable function

The components of a neural network

The solution is to use the **multilayer perceptron (MLP)**, or what we call now the neural networks

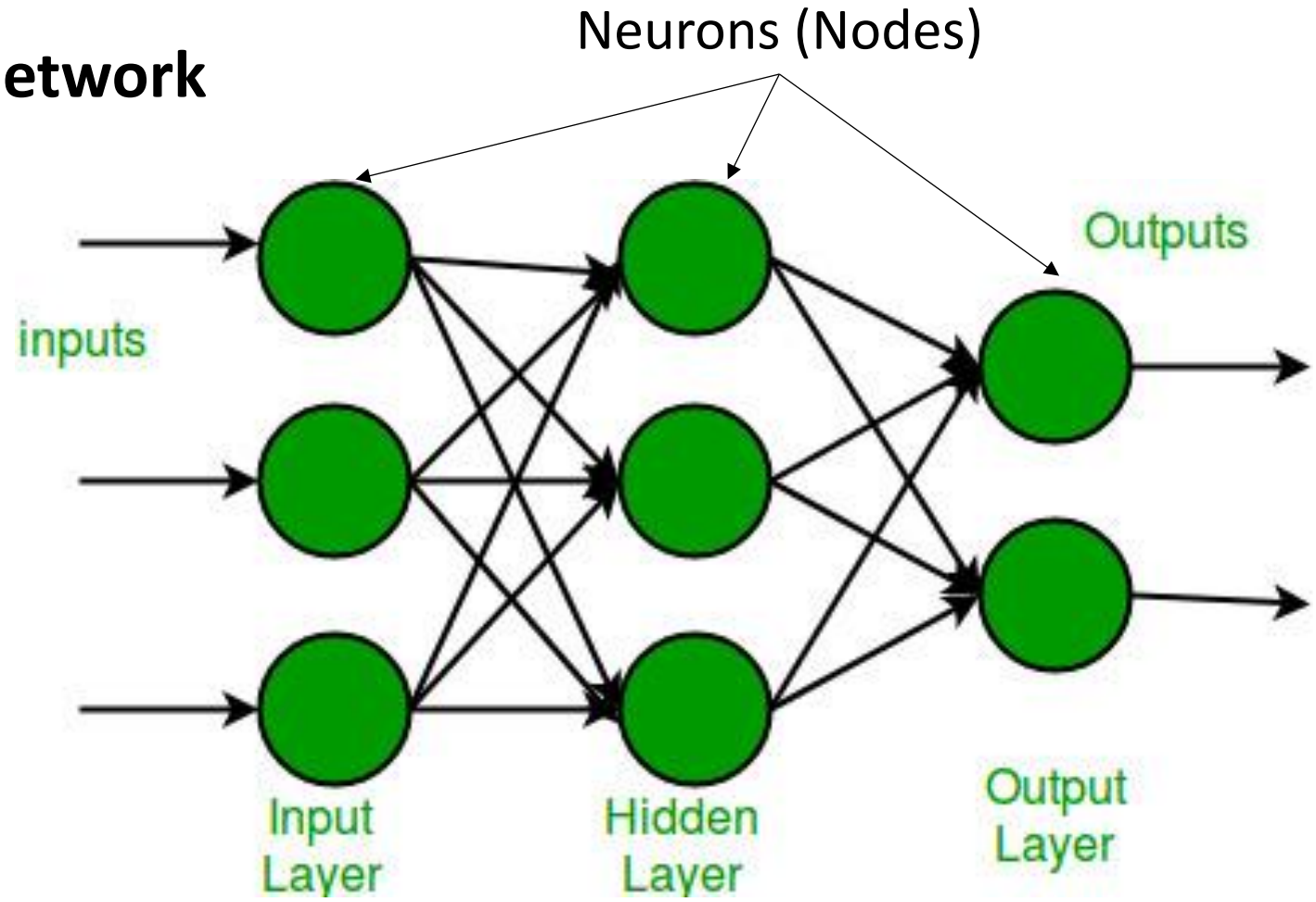


1. Neurons (Nodes)

The basic computational unit in a neural network

The components of a neural network

The solution is to use the **multilayer perceptron (MLP)**, or what we call now the neural networks

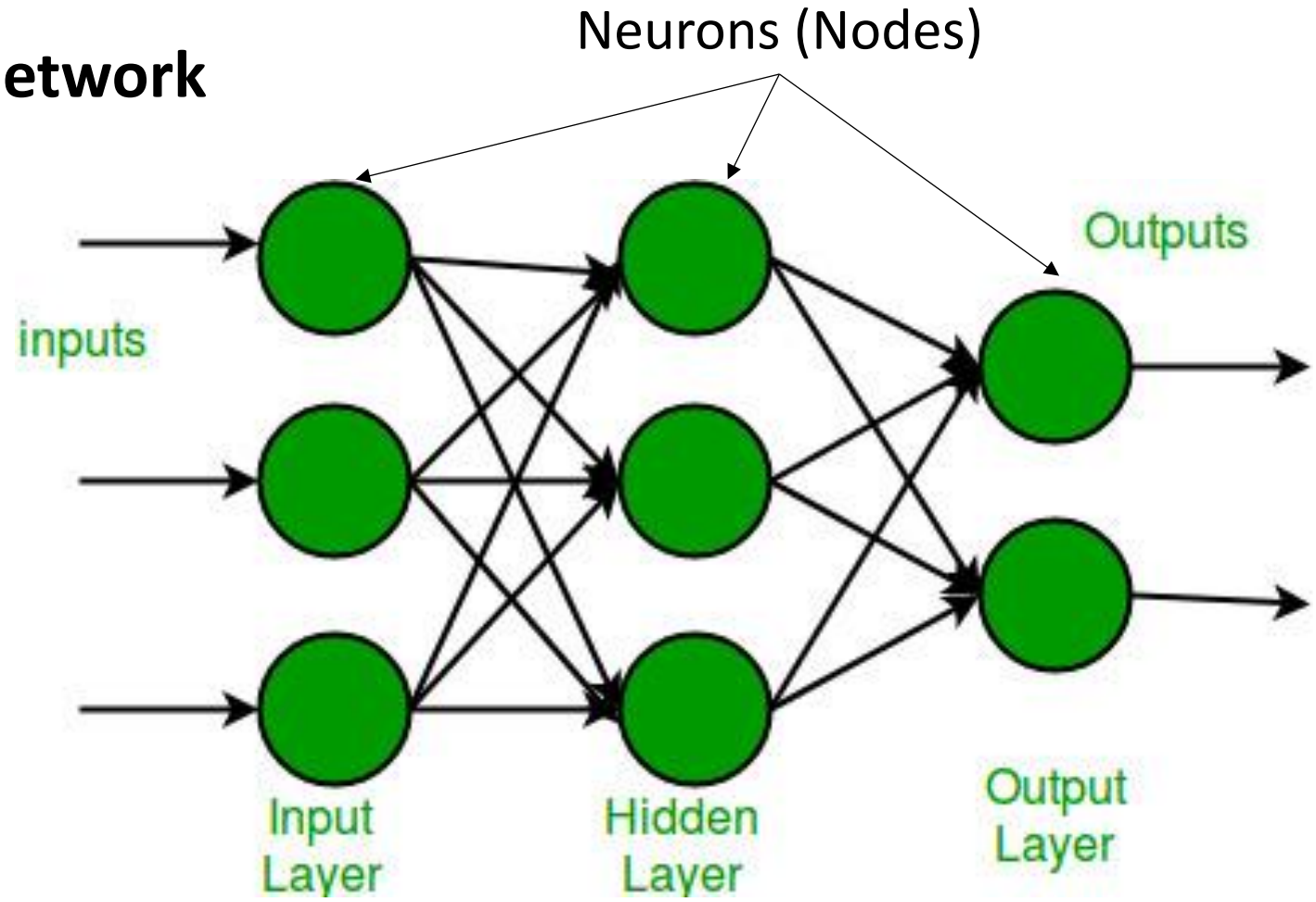


2. Layers

Neural networks are organized into layers:

The components of a neural network

The solution is to use the **multilayer perceptron (MLP)**, or what we call now the neural networks

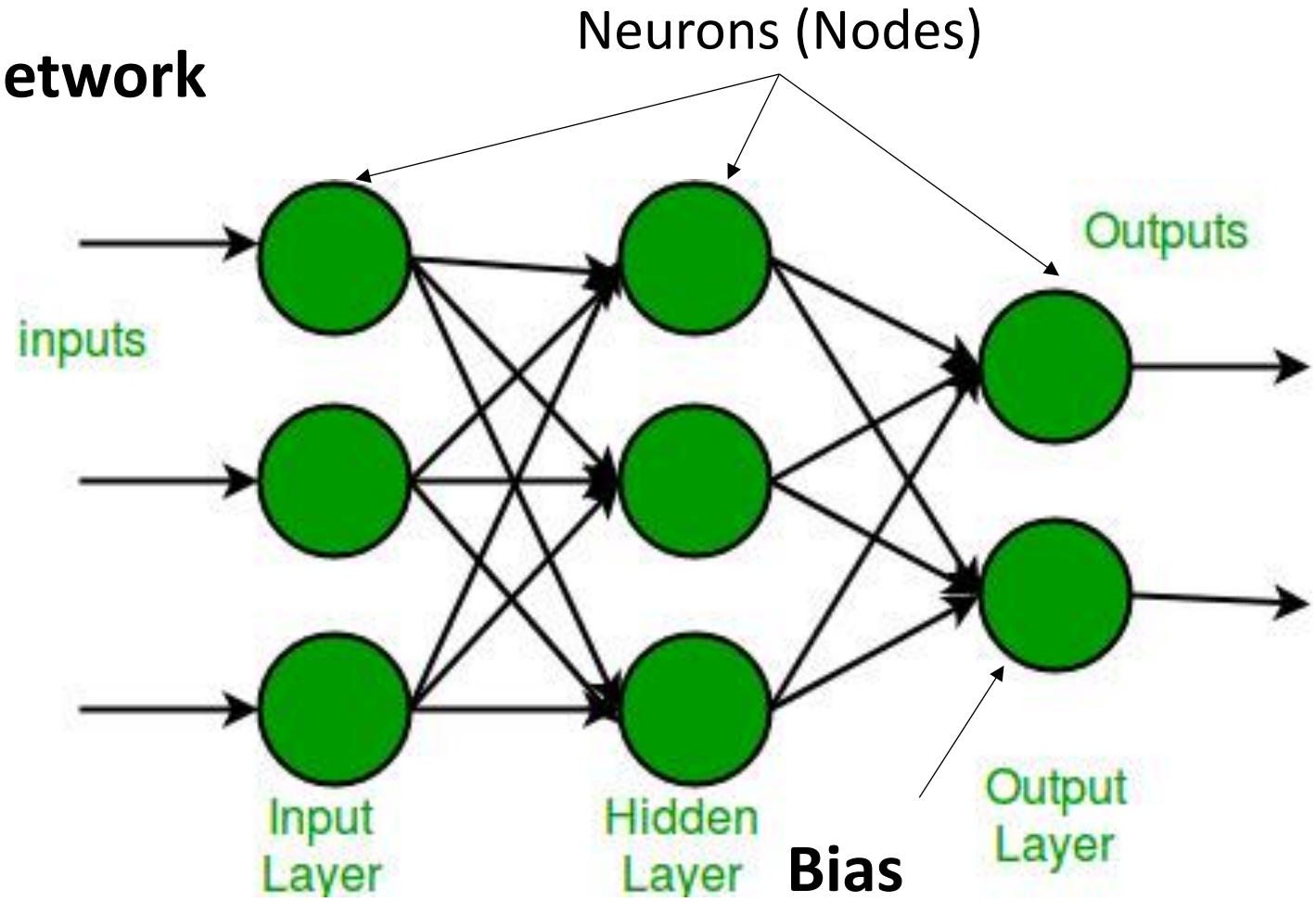


3. Weights

Parameters that determine the importance of each input connection to a neuron.

The components of a neural network

The solution is to use the **multilayer perceptron (MLP)**, or what we call now the neural networks

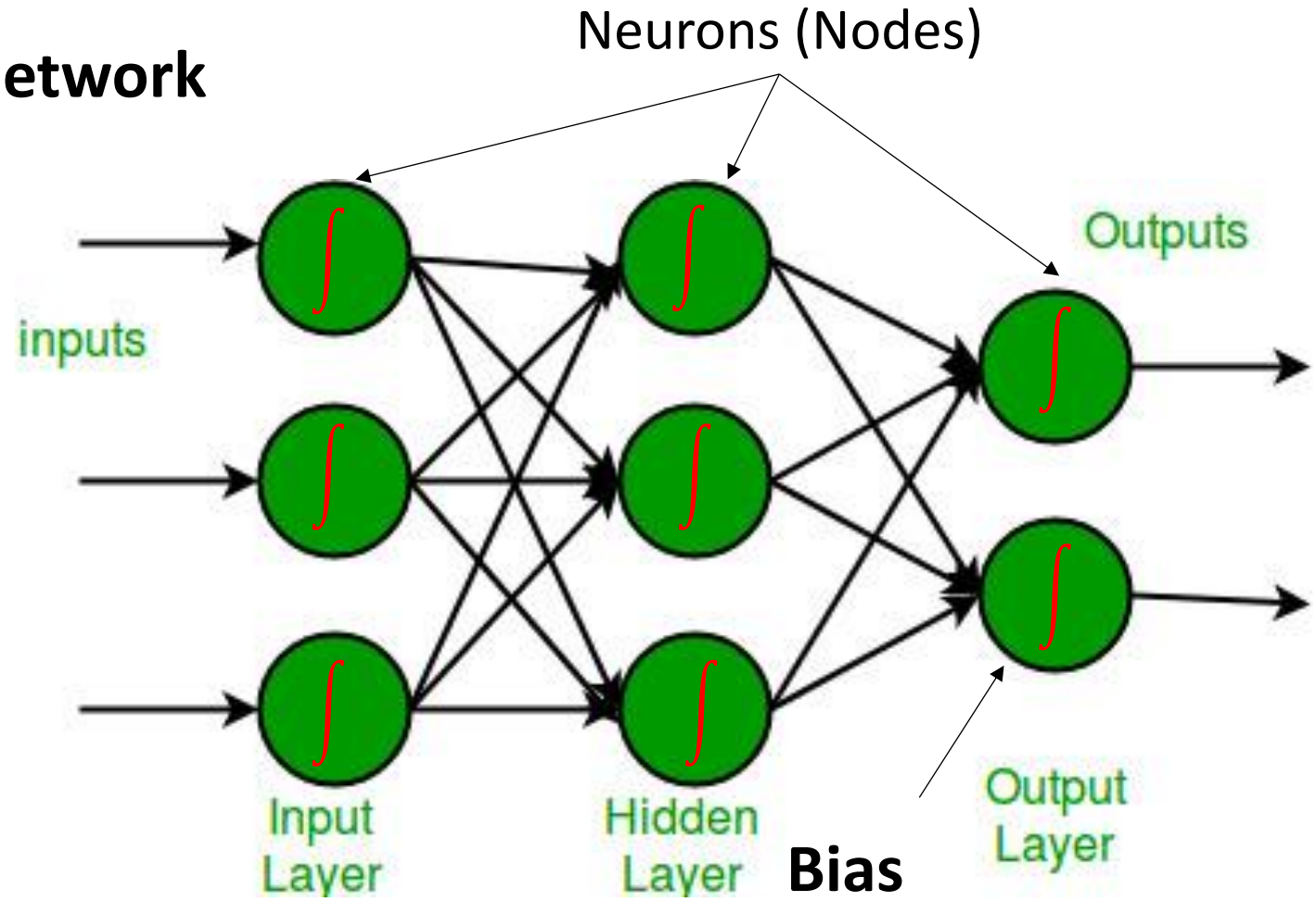


4. Bias

An additional parameter added to the weighted sum to shift the activation function's output.

The components of a neural network

The solution is to use the **multilayer perceptron (MLP)**, or what we call now the neural networks



5. Activation Functions

Mathematical functions applied to the weighted sum of inputs.

5. Activation Functions

Mathematical functions applied to the weighted sum of inputs.

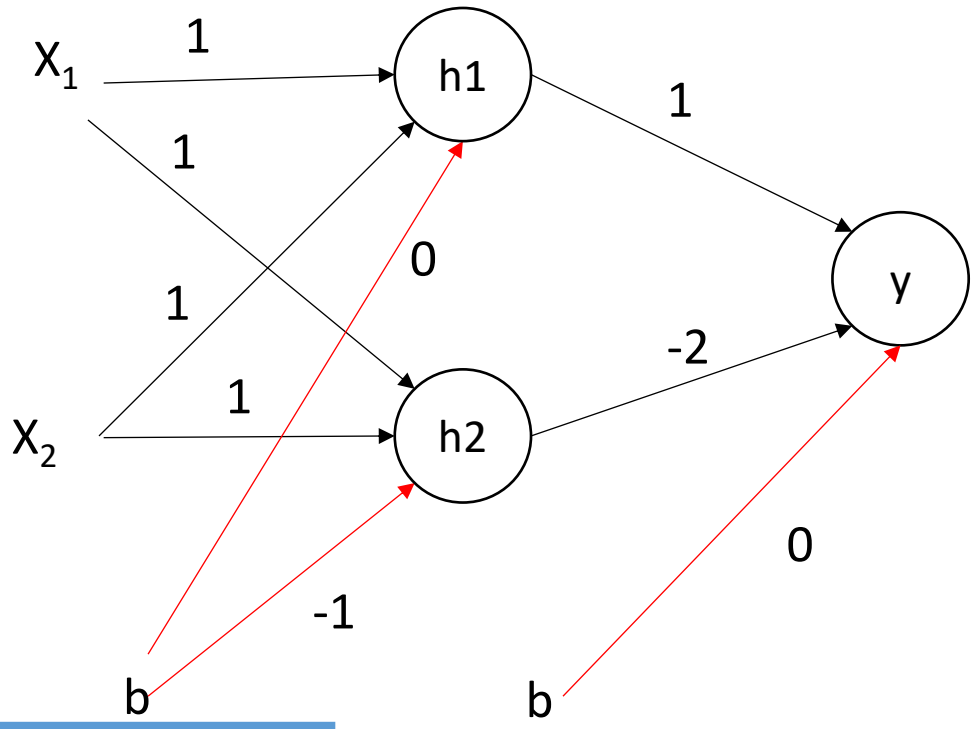
- **Sigmoid**: Outputs values between 0 and 1.

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

- **ReLU (Rectified Linear Unit):** Outputs max(0,x)

$$\mathrm{ReLU}(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0, \\ 0 & x \leq 0 \end{cases}$$

- **SoftPlus Function** $f(x) = \log(1 + e^x)$



XOR		
X1	X2	y
0	0	0
1	0	1
0	1	1
1	1	0

For 0,0

h1 = ReLU(x1*1+x2*1+0) = ReLU(0) = 0

h2 = ReLU(x1*1+x2*1-1) = ReLU(-1) = 0

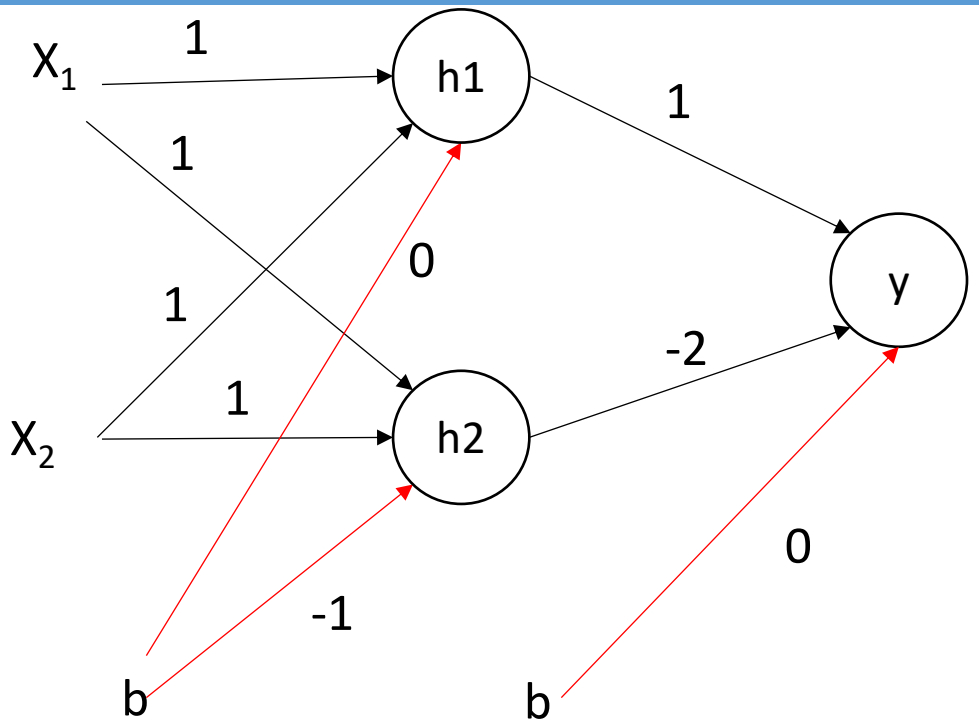
y = ReLU(h1*1+h2*-2+0) = ReLU(0) = 0

For 1,0

h1 = ReLU(x1*1+x2*1+0) = ReLU(1) = 1

h2 = ReLU(x1*1+x2*1-1) = ReLU(0) = 0

y = ReLU(h1*1+h2*-2+0) = ReLU(1) = 1



XOR			
X1	X2	y	h1
0	0	0	0
1	0	1	1
0	1	1	1
1	1	0	1

For 0,1

h1 = ReLU(x1*1+x2*1+0) = ReLU(1) = 1

h2 = ReLU(x1*1+x2*1-1) = ReLU(0) = 0

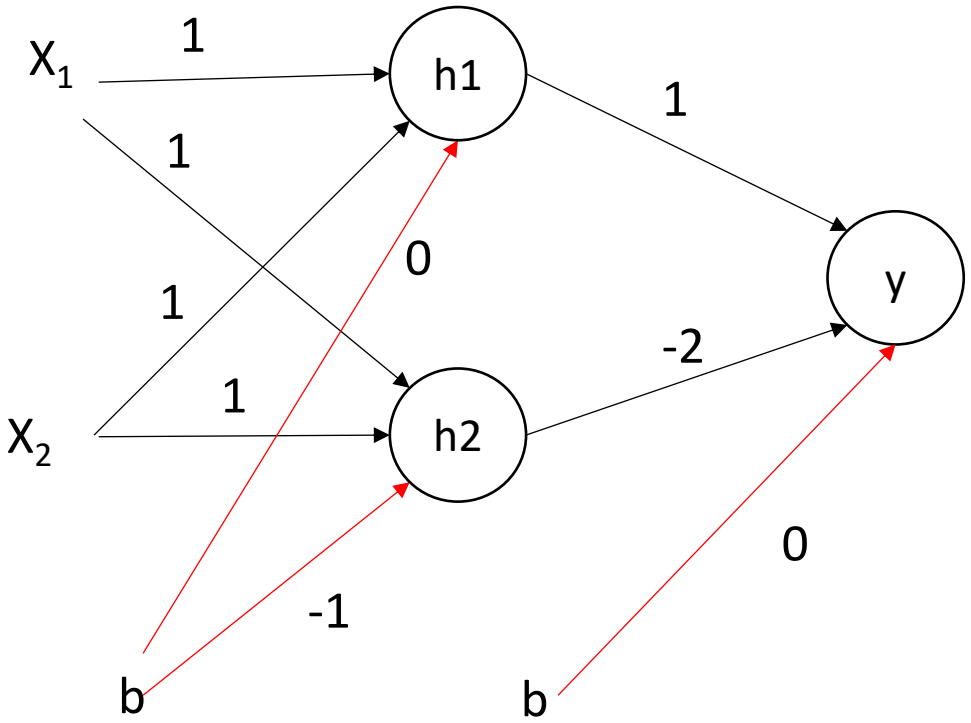
y = ReLU(h1*1+h2*-2+0) = ReLU(1) = 1

For 1,1

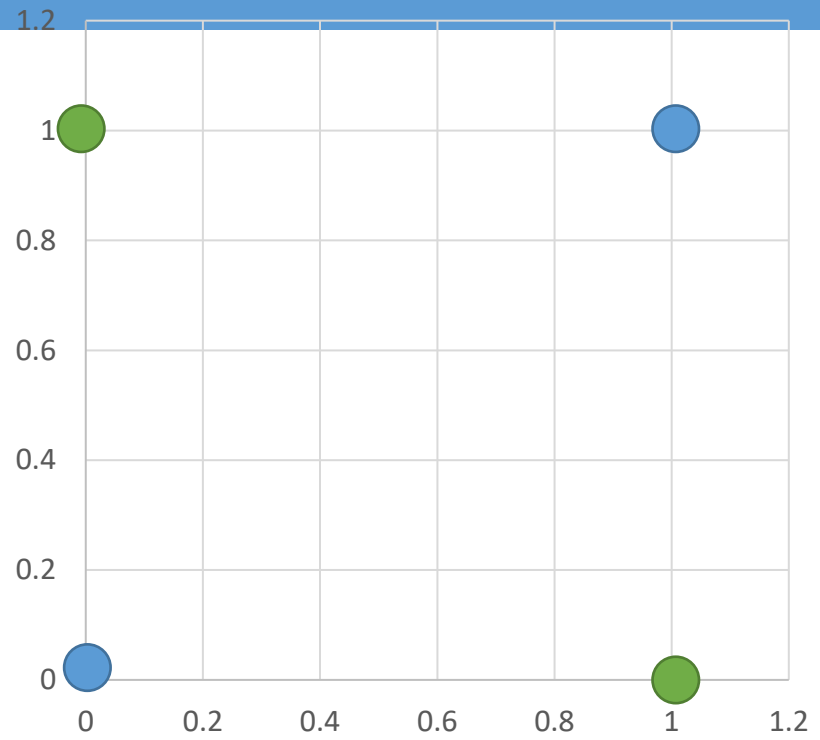
h1 = ReLU(x1*1+x2*1+0) = ReLU(2) = 2

h2 = ReLU(x1*1+x2*1-1) = ReLU(1) = 1

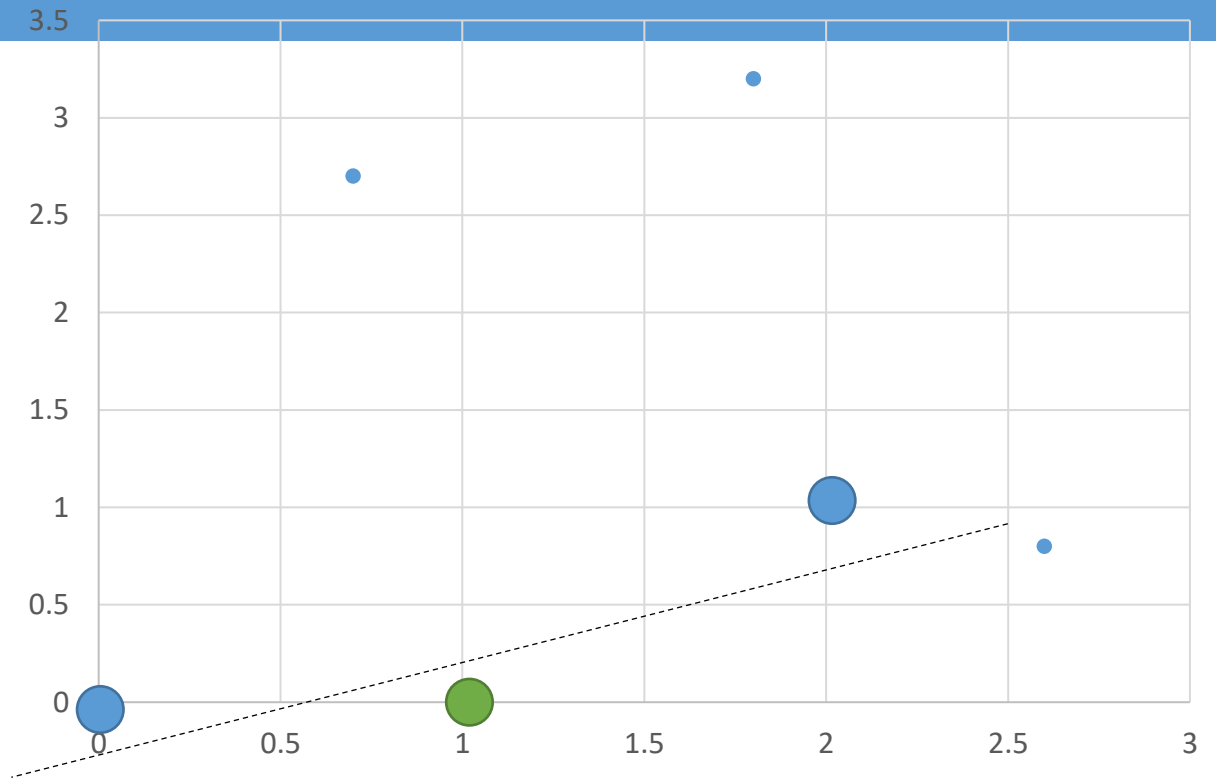
y = ReLU(h1*1+h2*-2+0) = ReLU(0) = 0



XOR					
X1	X2	y	h1	h2	y
0	0	0	0	0	0
1	0	1	1	0	1
0	1	1	1	0	1
1	1	0	2	1	0



The space of X



The space of h

XOR					
X1	X2	y	h1	h2	y
0	0	0	0	0	0
1	0	1	1	0	1
0	1	1	1	0	1
1	1	0	2	1	0

The chain rule

What is the derivation of the function y respect to x : $y = (3x^2 + 2)^5$

$$Y= 32+240x^2+720x^4+1080x^6+810x^8+243x^{10}.$$

The **chain rule** is a fundamental concept in calculus used to compute the derivative of a composite function. It allows you to calculate how a change in an input variable propagates t The **chain rule** through a sequence of dependent functions.

Mathematical Definition

If a function y depends on u, and u depends on x, i.e., $y = f(u)\ and\ u = g(x)$

Then the derivative of y with respect to x is:

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

This means you take the derivative of the outer function with respect to its argument (u) and multiply it by the derivative of the inner function (u) with respect to x.

The chain rule Example $y = (3x^2 + 2)^5$

Let's put y = u^5, and u = $(3x^2 + 2)$

According to the chain rule : $\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx}$

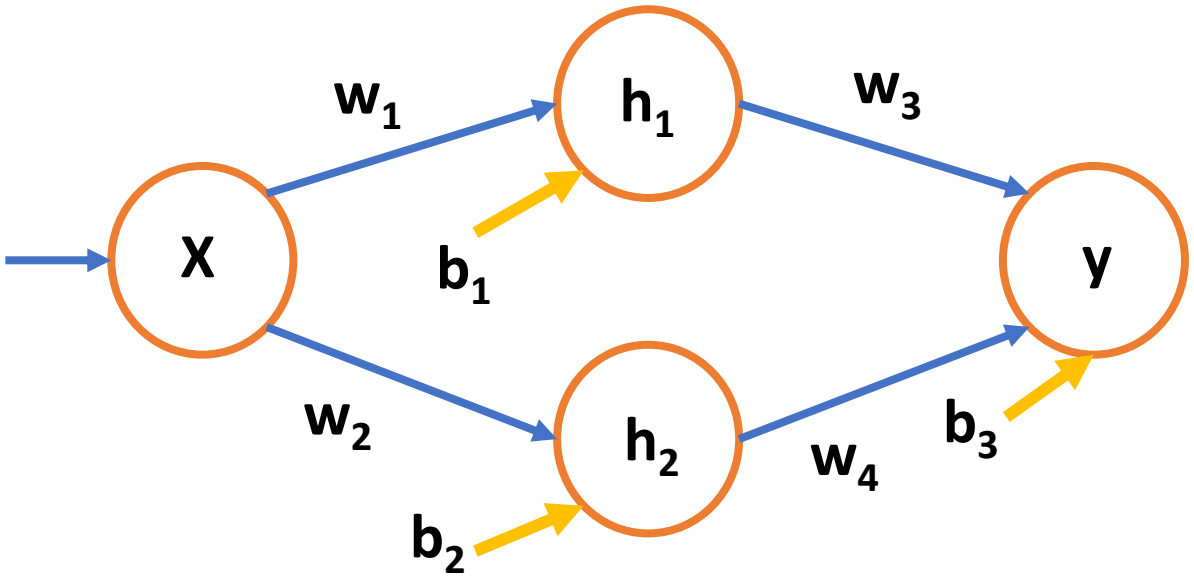
$$\frac{dy}{dx} = \frac{du^5}{du} * \frac{d(3x^2 + 2)}{dx}$$

$5u^4 *$ $6x$ Substitute: $5(3x^2 + 2)^4 * 6x = 30x(3x^2 + 2)^4$

Neural Network Example

Let's consider the following dataset which represents the value of any given function.

Input	Real observed
0	0
0.5	1
1	0

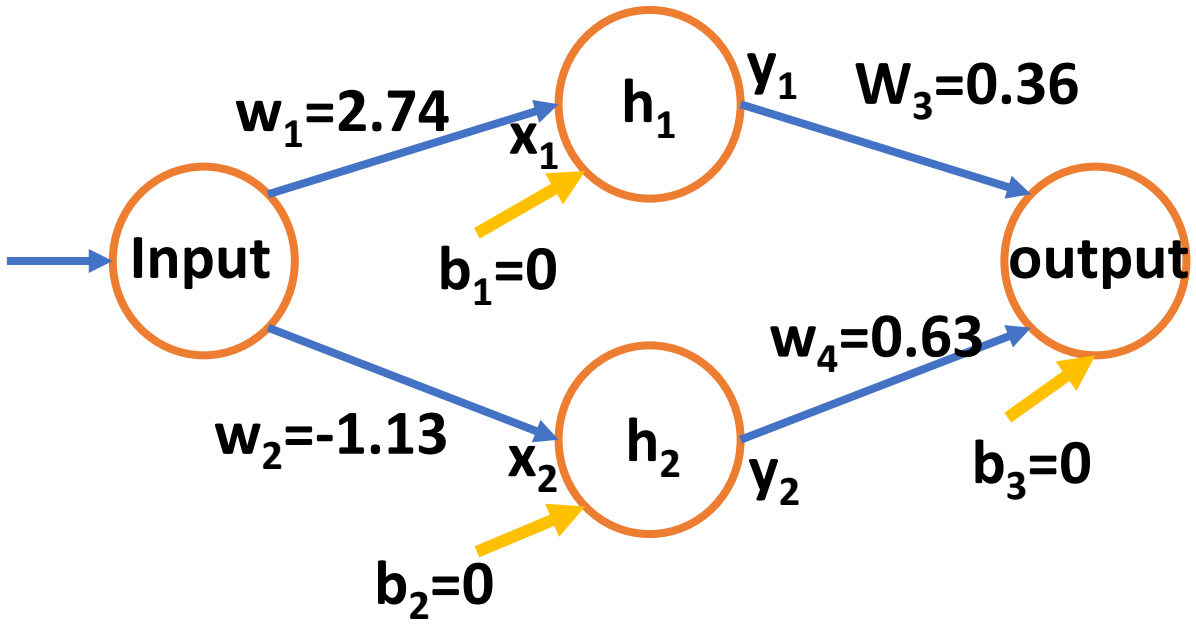


Initializing Weights and biases

In a neural network, the initial values for the **weights** (w) and **biases** (b) are crucial for starting the learning process. Proper initialization helps the model learn effectively and avoids issues like vanishing gradients or slow convergence

- **Uniform Distribution**: Weights are randomly initialized in a uniform range (e.g., [−3,3]).

- **Biases are initialized to 0**



Activation Functions

We use the **SoftPlus Function**

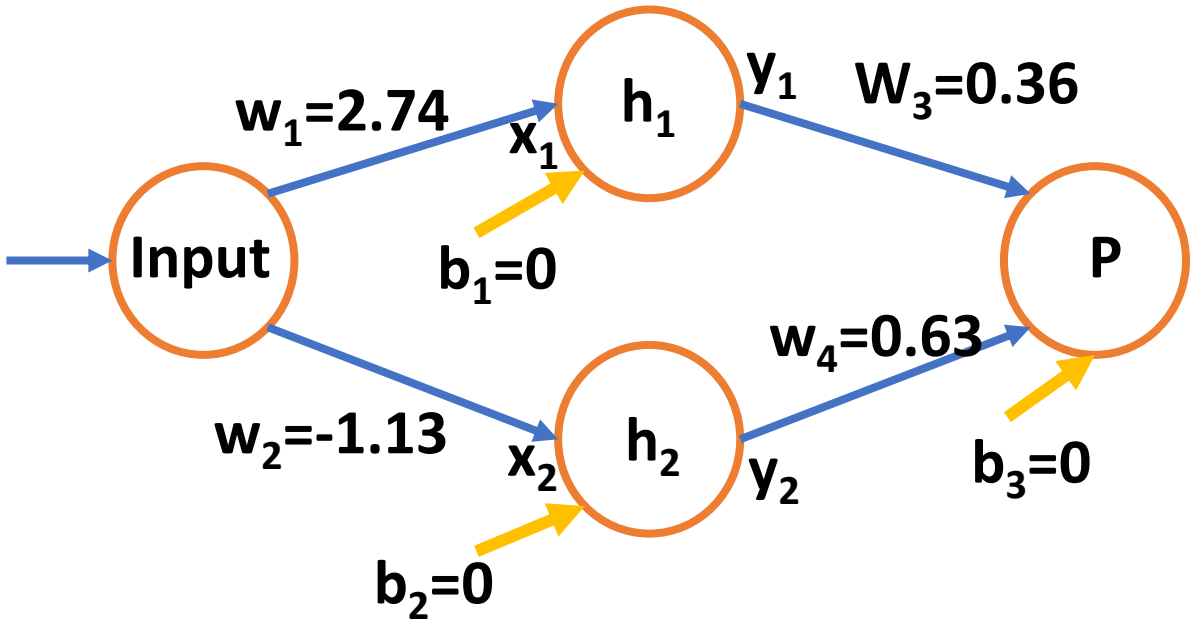
$$f(x) = \log(1 + e^x)$$

Activation Functions

We use the **SoftPlus** activation function defined in the previous slides. $f(x) = \log(1 + e^x)$

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_1 + b_2 \qquad y_2 = f(x_2)$$

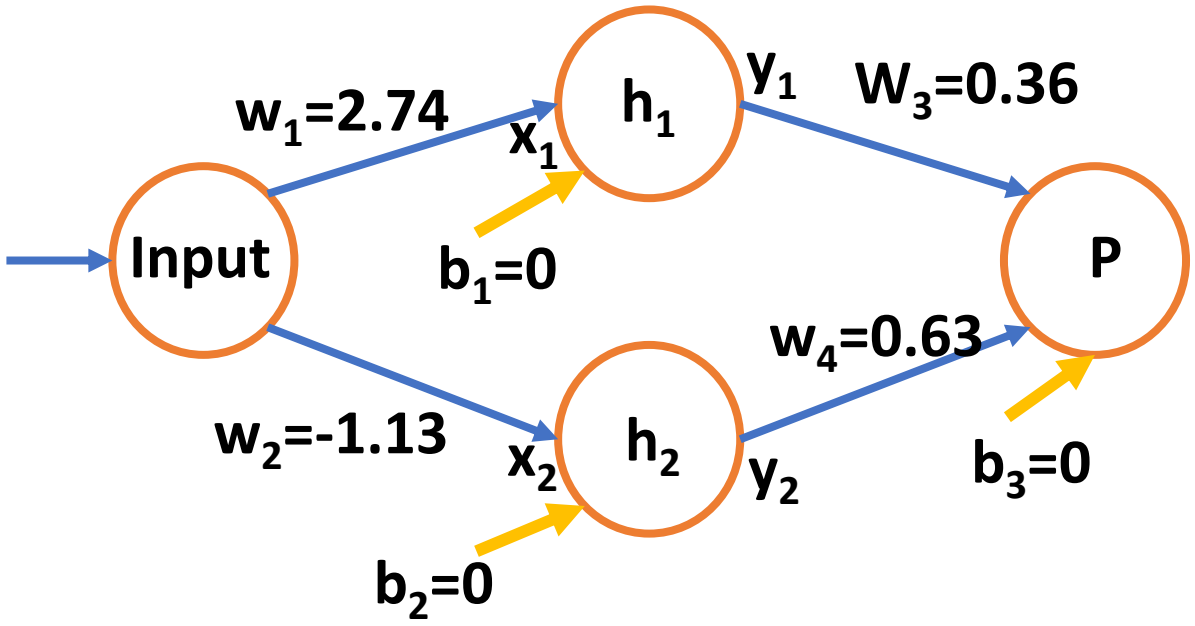
$$P = y_1 * w_3 + y_2 * w_4 + b_3$$



Calculation of the predicted values for the first row

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_2 + b_2 \qquad y_2 = f(x_2)$$

$$P = y_1 * w_3 + y_2 * w_4 + b_3$$

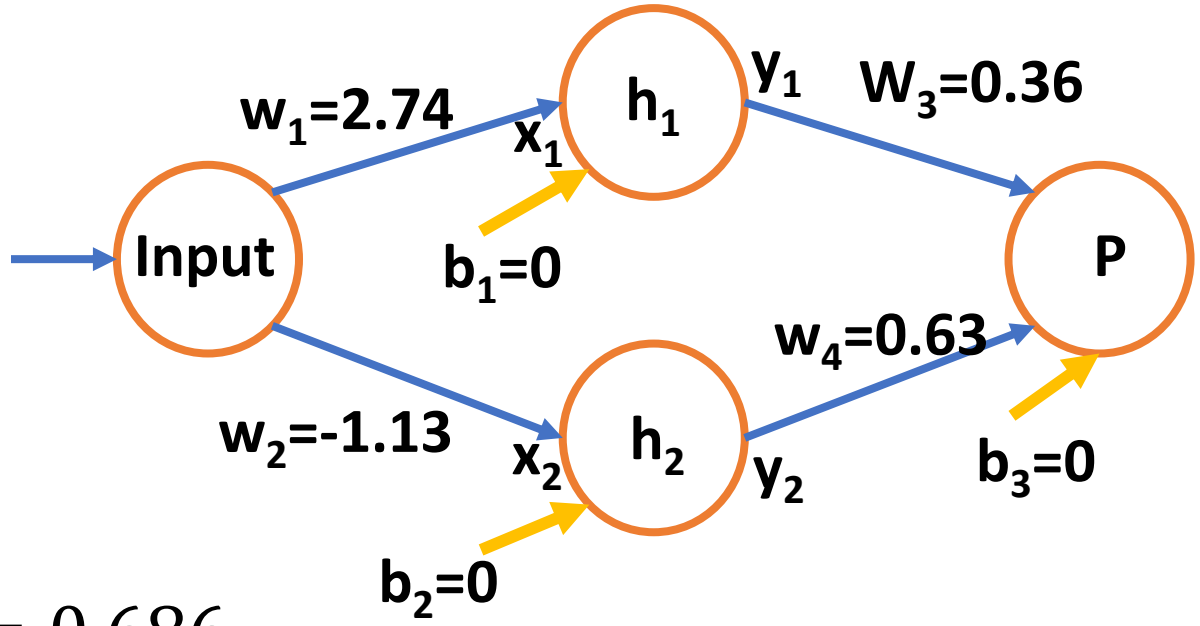


Calculation of the predicted values for the first row

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_2 + b_2 \qquad y_2 = f(x_2)$$

$$P = y_1 * w_3 + y_2 * w_4 + b_3$$

Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0					
0.5	1					0.85
1	0					1.19



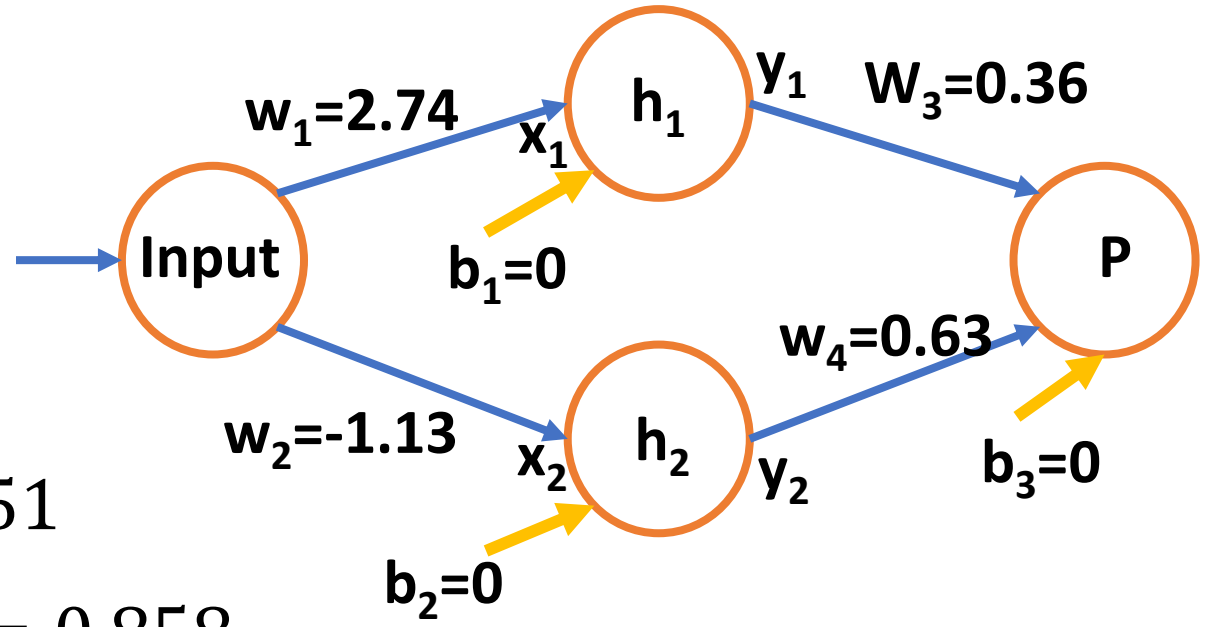
$$x_1 = 0 * 2.74 + 0 = 0$$
$$x_2 = 0 * (-1.13) + 0 = 0$$
$$y_1 = f(x_1) = \log(1 + e^0) = 0.693$$
$$y_2 = f(x_2) = \log(1 + e^0) = 0.693$$
$$P = 0.693 * 0.36 + 0.693 * 0.63 + 0 = 0.686$$

Calculation of the predicted values for the second row

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_2 + b_2 \qquad y_2 = f(x_2)$$

$$P = y_1 * w_3 + y_2 * w_4 + b_3$$

Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1					0.85
1	0					1.19



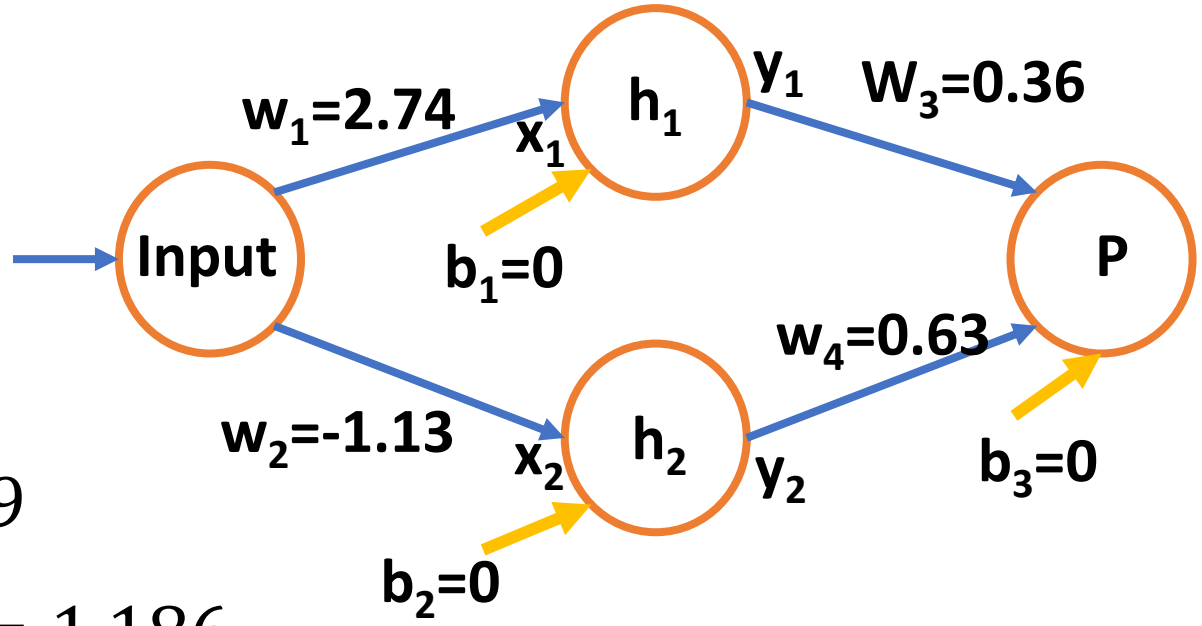
$$x_1 = 0.5 * 2.74 + 0 = 1.37$$
$$x_2 = 0.5 * (-1.13) + 0 = -0.565$$
$$y_1 = f(x_1) = \log(1 + e^{1.37}) = 1.598$$
$$y_2 = f(x_2) = \log(1 + e^{-0.565}) = 0.451$$
$$P = 1.598 * 0.36 + 0.451 * 0.63 + 0 = 0.858$$

Calculation of the predicted values for the third row

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_2 + b_2 \qquad y_2 = f(x_2)$$

$$P = y_1 * w_3 + y_2 * w_4 + b_3$$

Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0					1.19

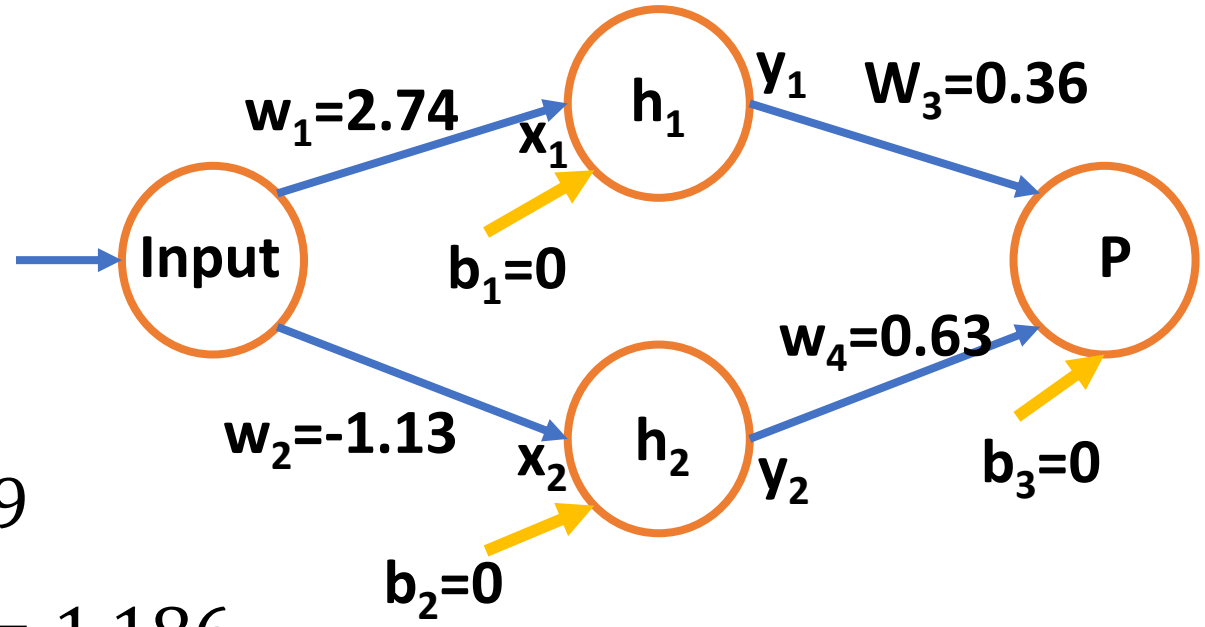


$$x_1 = 1 * 2.74 + 0 = 2.74$$
$$x_2 = 1 * (-1.13) + 0 = -1.13$$
$$y_1 = f(x_1) = \log(1 + e^{2.74}) = 2.804$$
$$y_2 = f(x_2) = \log(1 + e^{-1.13}) = 0.279$$
$$P = 2.804 * 0.36 + 0.279 * 0.63 + 0 = 1.186$$

Calculation of the predicted values for the first row

$$x_1 = input * w_1 + b_1 \qquad y_1 = f(x_1)$$
$$x_2 = input * w_2 + b_2 \qquad y_2 = f(x_2)$$
$$P = y_1 * w_3 + y_2 * w_4 + b_3$$

Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186

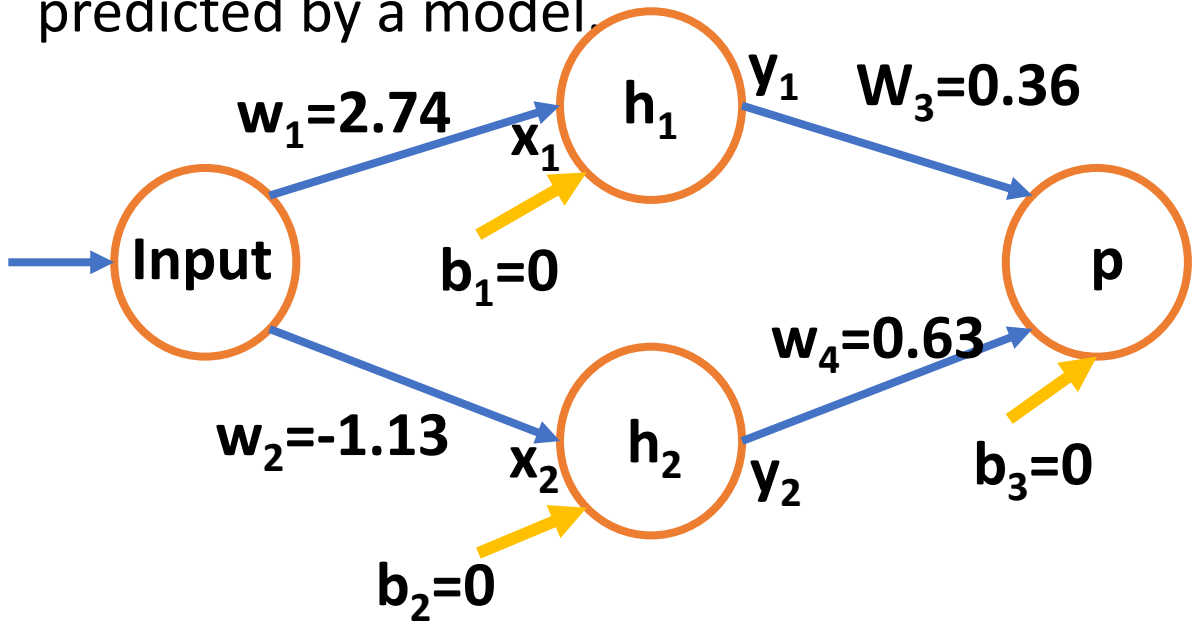


$$x_1 = 1 * 2.74 + 0 = 2.74$$
$$x_2 = 1 * (-1.13) + 0 = -1.13$$
$$y_1 = f(x_1) = \log(1 + e^{2.74}) = 2.804$$
$$y_2 = f(x_2) = \log(1 + e^{-1.13}) = 0.279$$
$$P = 2.804 * 0.36 + 0.279 * 0.63 + 0 = 1.186$$

The backpropagation algorithm

The first thing that we have to de is to measure the error

The **Residual Sum of Squares (RSS)** is a measure used in regression analysis to evaluate the difference between the observed values and the values predicted by a model.



Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186

$$\text{RSS} = \sum_{i=1}^{n}(O - P)^2$$

O is the actual value (the real value)

P is the predicted value for the i–th observation

n is the total number of observations.

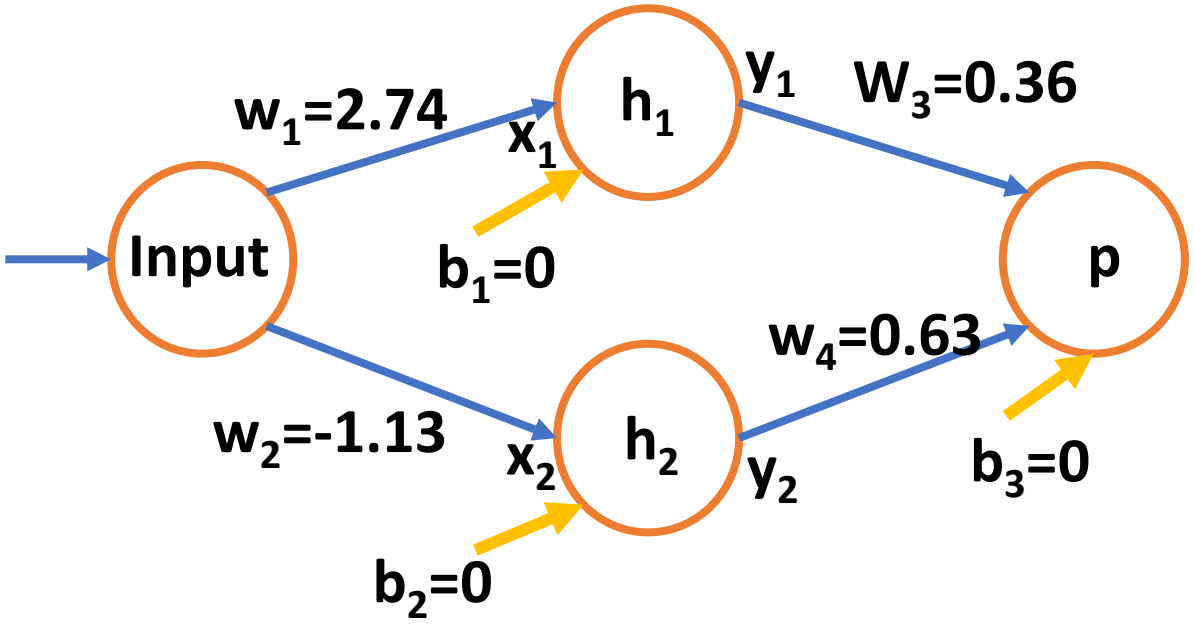
The backpropagation algorithm

We have to update the values of weights and biases

To do that we need to compute :

$$\frac{dRSS}{db_1} \qquad \frac{dRSS}{db_2} \qquad \frac{dRSS}{db_3}$$

Input	real	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



$$\frac{dRSS}{dw_1} \qquad \frac{dRSS}{dw_2}$$

$$\frac{dRSS}{dw_3} \qquad \frac{dRSS}{dw_4}$$

$$\text{RSS} = \sum_{i=1}^{n} (O - P)^2$$

The backpropagation algorithm

$$\frac{dRSS}{db_1} = \frac{dRSS}{dP} * \frac{dP}{dy_1} * \frac{dy_1}{dx_1} * \frac{dx_1}{db_1}$$

$$\frac{dRSS}{dP} = \frac{d\sum_{i=1}^{3}(O-P)^2}{dP} = \frac{\sum_{i=1}^{3} d(O-P)^2}{dP}$$

$$\frac{dP}{dy_1} = \frac{d(y_1 * w_3 + y_2 * w_4 + b_3)}{dy_1}$$

$$\frac{dy_1}{dx_1} = \frac{d\log(1+e^{x_1})}{dx_1} = \frac{e^{x_1}}{1+e^{x_1}}$$

$$\frac{dx_1}{db_1} = \frac{d(input * w_1 + b_1)}{db1} = 1$$

$$\frac{dRSS}{dP} = \sum_{i=1}^{3} -2(O-P)$$

$$\frac{dP}{dy_1} = w_3 \qquad \frac{dy_1}{dx_1} = \frac{e^{x_1}}{1+e^{x_1}}$$

$$\frac{dx_1}{db_1} = 1$$

$$\text{RSS} = \sum_{i=1}^{n}(O-P)^2$$

The backpropagation algorithm

$$\frac{dRSS}{db_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * 1$$

$$\frac{dRSS}{db_2} = \sum_{i=1}^{3} -2(O_i - P_i) * w_4 * \frac{e^{x_2}}{1+e^{x_2}} * 1$$

$$\frac{dRSS}{db_3} = \frac{dRSS}{dP} * \frac{dP}{db_3} \qquad \frac{dP}{db_3} = \frac{d(y_1 * w_3 + y_2 * w_4 + b_3)}{db_3}$$

$$= \sum_{i=1}^{3} -2(O_i - P_i) * 1$$

The backpropagation algorithm

$$\frac{dRSS}{dw_1} = \frac{dRSS}{dP} * \frac{dP}{dy_1} * \frac{dy_1}{dx_1} * \frac{dx_1}{dw_1}$$

$$\frac{dRSS}{dP} = \sum_{i=1}^{3} -2(O - P)$$

$$\frac{dP}{dy_1} = \frac{d(y_1 * w_3 + y_2 * w_4 + b_3)}{dy_1} \qquad \frac{dP}{dy_1} = w_3$$

$$\frac{dy_1}{dx_1} = \frac{d \log(1 + e^{x_1})}{dx_1} = \frac{e^{x_1}}{1 + e^{x_1}}$$

$$\frac{dx_1}{dw_1} = \frac{d(input * w_1 + b_1)}{db1} = input$$

$$\frac{dRSS}{dw_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * \text{input}$$

The backpropagation algorithm

$$\frac{dRSS}{dw_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * \text{input}_i$$

$$\frac{dRSS}{dw_2} = \sum_{i=1}^{3} -2(O_i - P_i) * w_4 * \frac{e^{x_2}}{1+e^{x_2}} * \text{input}_i$$

$$\frac{dRSS}{dw_3} = \frac{dRSS}{dP} * \frac{dP}{dw_3} \qquad \frac{dP}{dw_3} = \frac{d(y_1 * w_3 + y_2 * w_4 + b_3)}{dw_3} = y_1$$

$$\frac{dRSS}{dw_3} = \sum_{i=1}^{3} -2(O_i - P_i) * y_1$$

$$\frac{dRSS}{dw_4} = \sum_{i=1}^{3} -2(O_i - P_i) * y_2$$

The backpropagation algorithm

$$\frac{dRSS}{dw_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * \text{input}_i$$

$$\frac{dRSS}{dw_2} = \sum_{i=1}^{3} -2(O_i - P_i) * w_4 * \frac{e^{x_2}}{1+e^{x_2}} * \text{input}_i$$

$$\frac{dRSS}{dw_3} = \sum_{i=1}^{3} -2(O_i - P_i) * \text{y}_1$$

$$\frac{dRSS}{dw_4} = \sum_{i=1}^{3} -2(O_i - P_i) * \text{y}_2$$

The backpropagation algorithm

$$\frac{dRSS}{db_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * 1$$

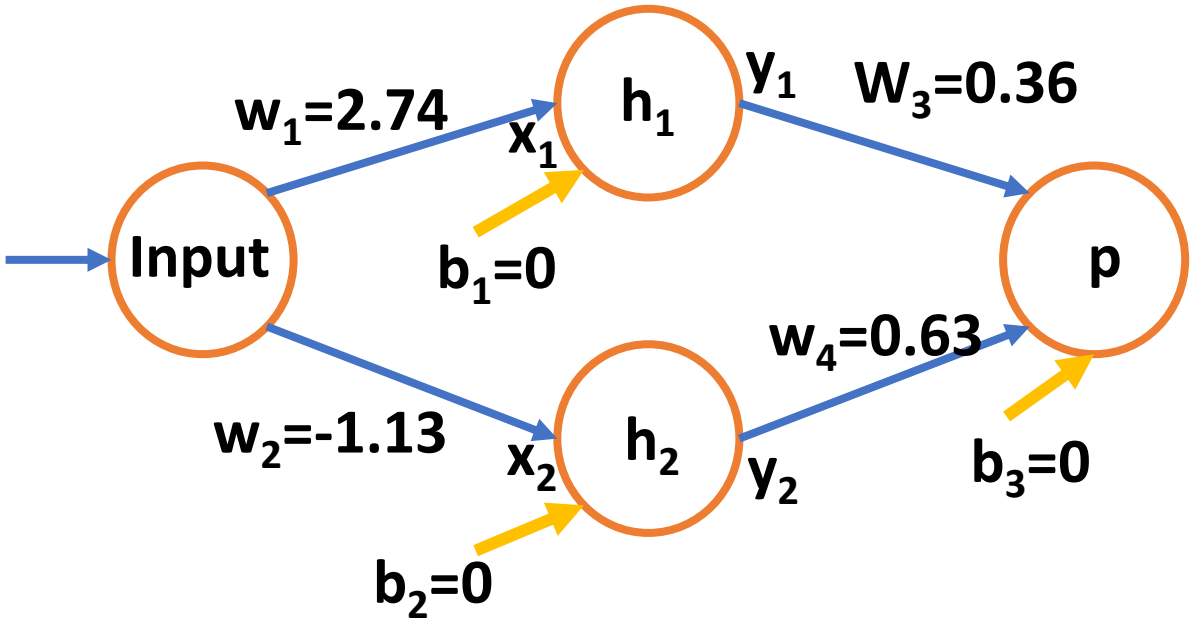
$$= -2(0 - 0.686) * 0.36 * \frac{e^{0}}{1+e^{0}}$$

$$-2(1 - 0.858) * 0.36 * \frac{e^{1.37}}{1 + e^{1.37}}$$

$$-2(0 - 1.186) * 0.36 * \frac{e^{2.74}}{1 + e^{2.74}}$$

$$= 0.965$$

Input	Observed	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



The backpropagation algorithm

$$\frac{dRSS}{db_2} = \sum_{i=1}^{3} -2(O_i - P_i) * w_4 * \frac{e^{x_2}}{1+e^{x_2}} * 1$$

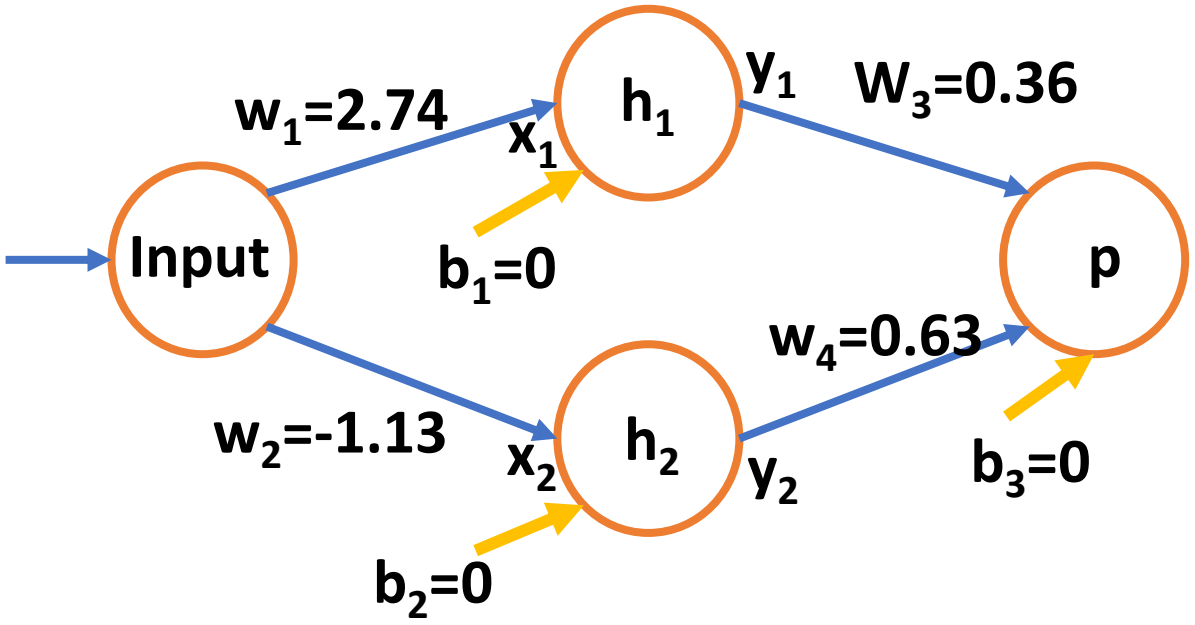
$$= -2(0 - 0.686) * 0.63 * \frac{e^{0}}{1+e^{0}}$$

$$-2(1 - 0.858) * 0.63 * \frac{e^{-0.565}}{1 + e^{-0.565}}$$

$$-2(0 - 1.186) * 0.63 * \frac{e^{-1.13}}{1 + e^{-1.13}}$$

$$= 0.740$$

Input	Observed	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



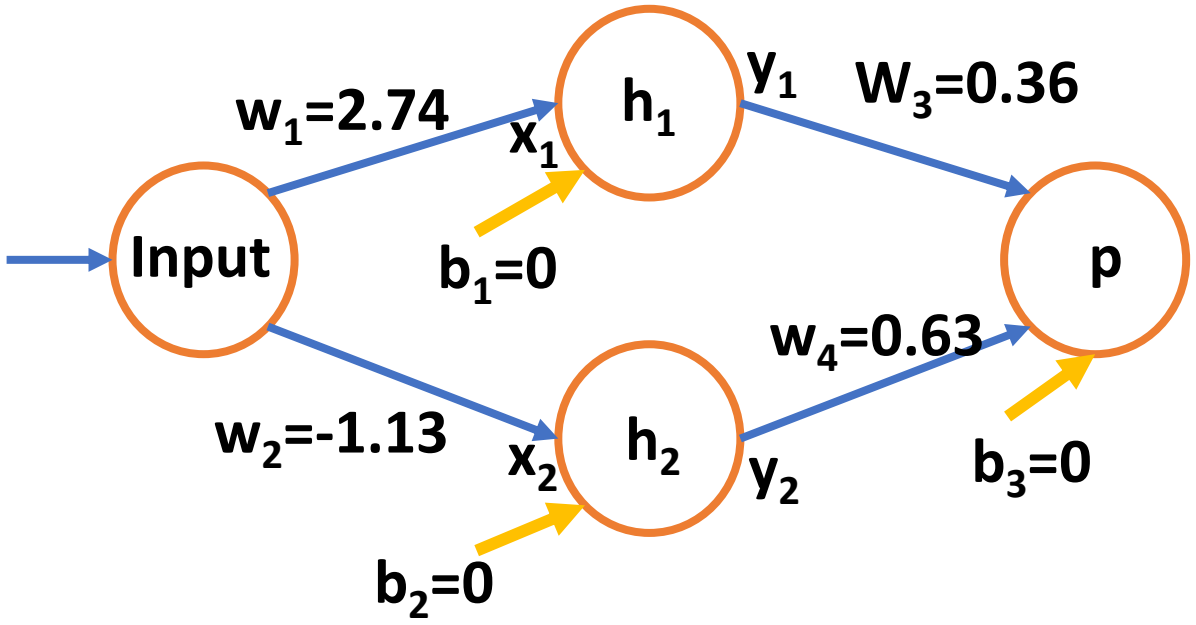
The backpropagation algorithm

$$\frac{dRSS}{db_3} = \sum_{i=1}^{3} -2(O_i - P_i) * 1$$

$$= -2(0 - 0.686) - 2(1 - 0.858) - 2(0 - 1.186)$$

$$= 3.460$$

Input	Observed	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



The backpropagation algorithm

$$\frac{dRSS}{dw_1} = \sum_{i=1}^{3} -2(O_i - P_i) * w_3 * \frac{e^{x_1}}{1+e^{x_1}} * \text{input}_i$$

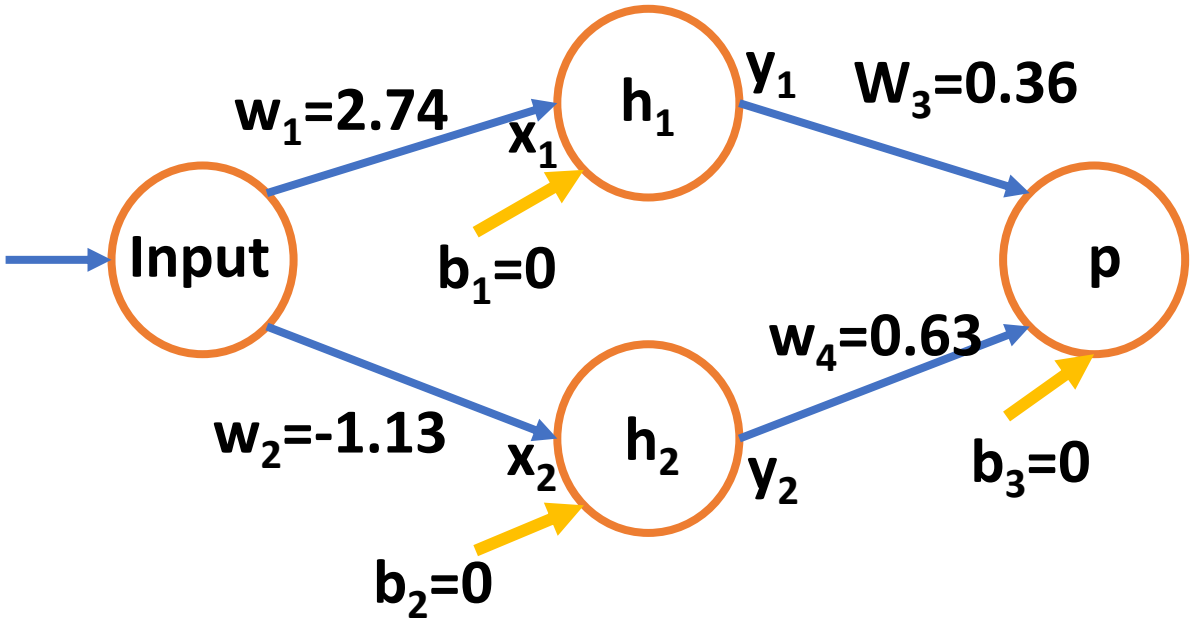
$$= -2(0 - 0.686) * 0.36 * \frac{e^{0}}{1+e^{0}} * 0$$

$$-2(1 - 0.858) * 0.36 * \frac{e^{1.37}}{1+e^{1.37}} * 0.5$$

$$-2(0 - 1.186) * 03.6 * \frac{e^{2.74}}{1+e^{2.74}} * 1$$

$$= 0.760$$

Input	Observed	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



The backpropagation algorithm

In the same way

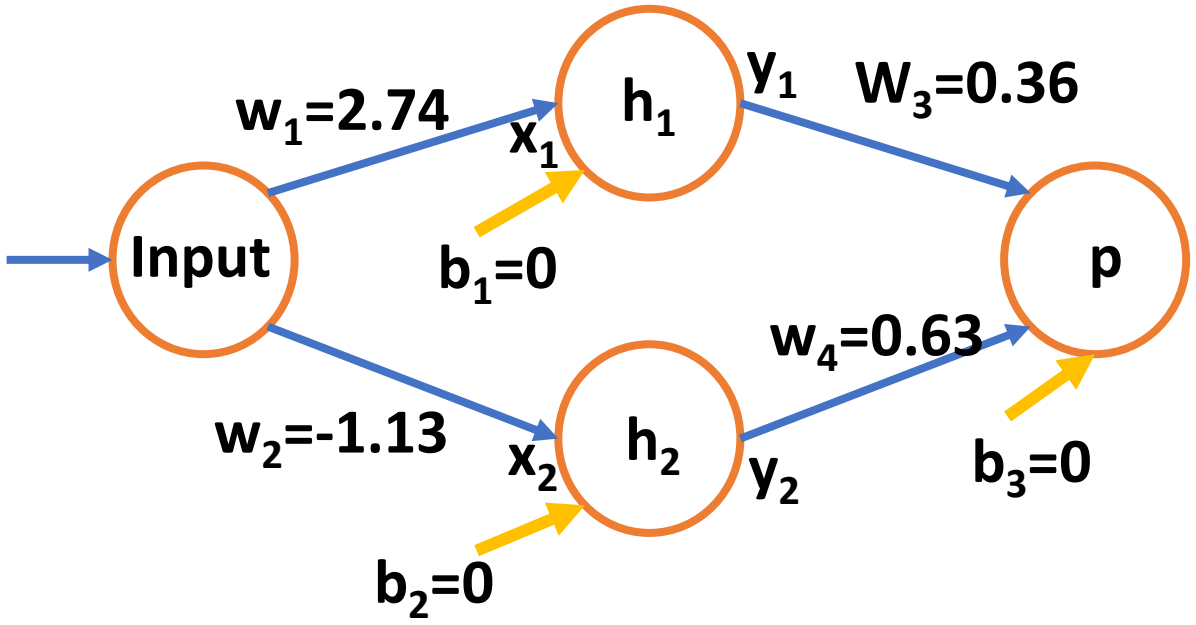
$$\frac{dRSS}{db_1} = 0.965 \qquad \frac{dRSS}{dw_1} = 0.760$$

$$\frac{dRSS}{db_2} = 0.740 \qquad \frac{dRSS}{dw_2} = 0.33$$

$$\frac{dRSS}{db_3} = 3.460 \qquad \frac{dRSS}{dw_3} = 7.14$$

$$\frac{dRSS}{dw_4} = 1.49$$

Input	Observed	x_1	x_2	y_1	y_2	Predicted values
0	0	0	0	0.693	0.693	0.686
0.5	1	1.37	-0.565	1.598	0.451	0.858
1	0	2.74	-1.13	2.804	0.279	1.186



New Values

Step = Derivation * Learning rate

New value = Old value - Step

Learning rate = 0.1

For b_1

$Step = 0.965 * 0.1 = 0.0965$

$New\ b_1 = 0 - 0.0965 = -0.0965$

For b_2

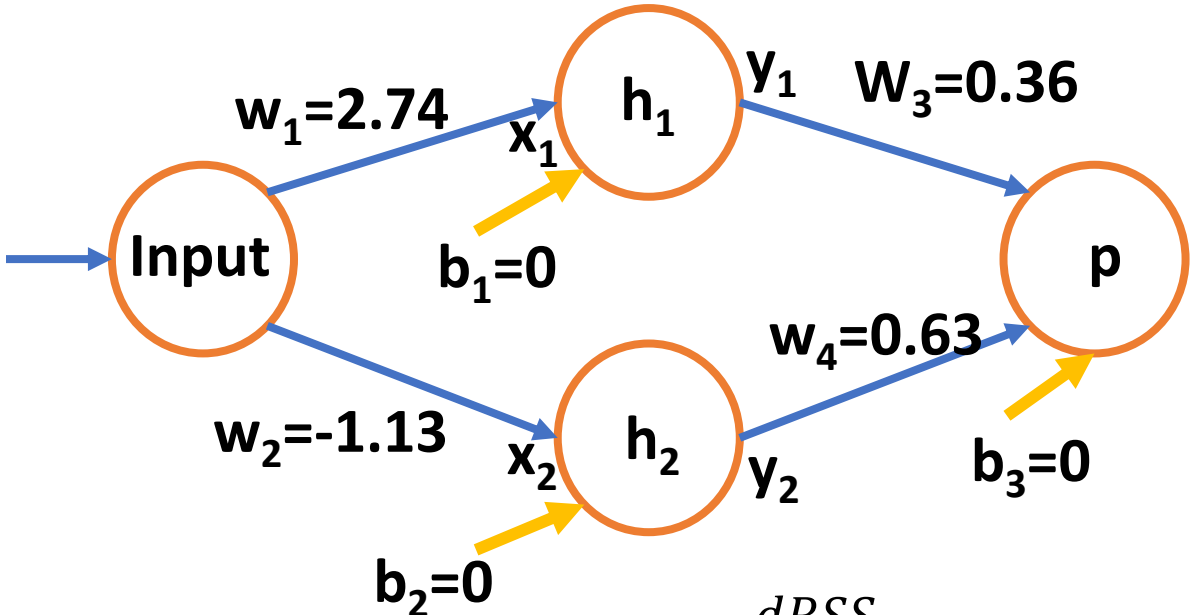
$Step = 0.740 * 0.1 = 0.074$

$New\ b_2 = 0 - 0.074 = -0.074$

For b_3

$Step = 3.460 * 0.1 = 0.346$

$New\ b_3 = 0 - 0.346 = -0.346$



$$\frac{dRSS}{db_1} = 0.965$$

$$\frac{dRSS}{db_2} = 0.740$$

$$\frac{dRSS}{db_3} = 3.460$$

$$\frac{dRSS}{dw_1} = 0.760$$

$$\frac{dRSS}{dw_2} = 0.33$$

$$\frac{dRSS}{dw_3} = 7.14$$

$$\frac{dRSS}{dw_4} = 1.49$$

New Values

Learning rate = 0.1

Step = Derivation * Learning rate

New value = Old value - Step

For w_1

$Step = 0.760 * 0.1 = 0.076$

$New\ w_1 = 2.74 - 0.076 = 2.664$

For w_2

$Step = 0.33 * 0.1 = 0.033$

$New\ w_2 = -1.13 - 0.033 = -1.163$

For w_3

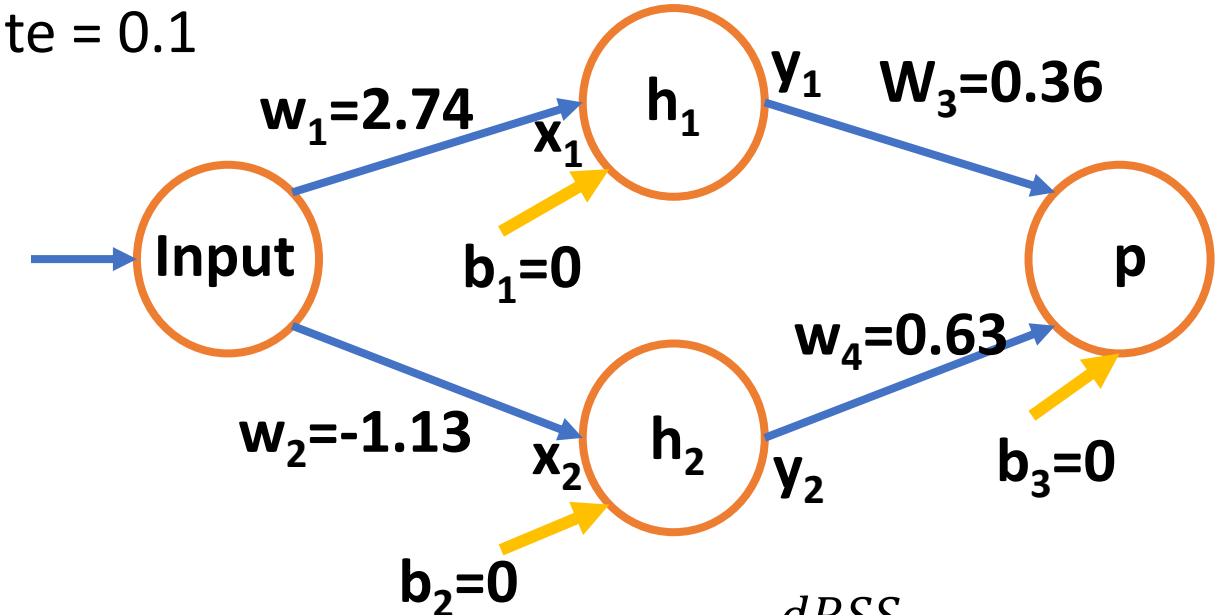
$Step = 7.14 * 0.1 = 0.714$

$New\ w_3 = 0.36 - 0.0714 = 0.2886$

For w_4

$Step = 1.49 * 0.1 = 0.149$

$New\ w_4 = 0.63 - 0.149 = 0.481$



$\frac{dRSS}{db_1} = 0.965$

$\frac{dRSS}{db_2} = 0.740$

$\frac{dRSS}{db_3} = 3.460$

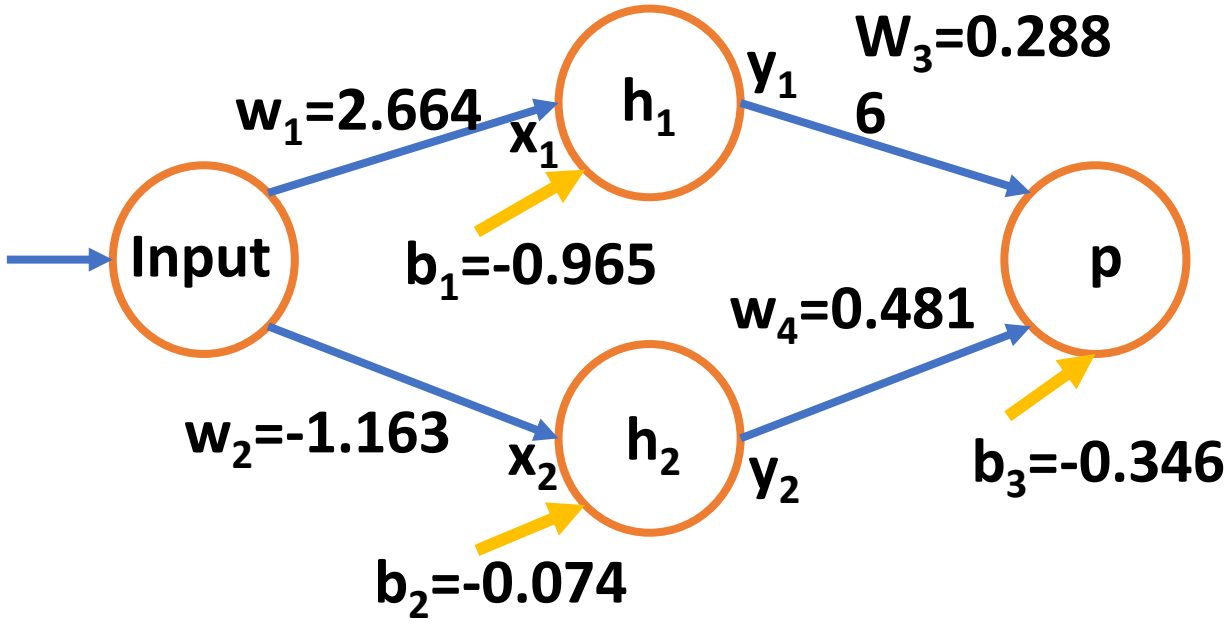
$\frac{dRSS}{dw_1} = 0.760$

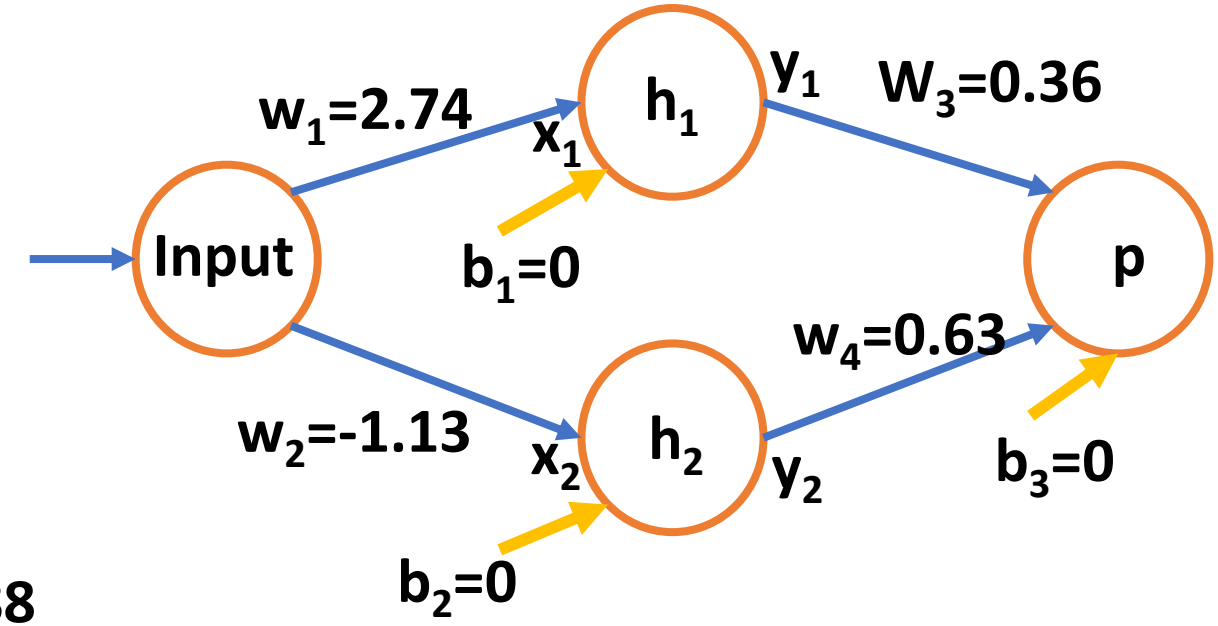
$\frac{dRSS}{dw_2} = 0.33$

$\frac{dRSS}{dw_3} = 7.14$

$\frac{dRSS}{dw_4} = 1.49$

New Values


Input
h1
h2
p
w1=2.664
x1
b1=-0.965
y1
W3=0.288
6
w2=-1.163
x2
b2=-0.074
y2
w4=0.481
b3=-0.346


Input
h1
h2
p
w1=2.74
x1
b1=0
y1
W3=0.36
w2=-1.13
x2
b2=0
y2
w4=0.63
b3=0