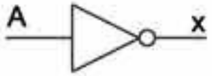
ECS 171 Machine Learning

Lecture6: Emulating Boolean Functions with ANN, Practice Example

Instructor: Dr. Setareh Rafatirad

Emulating Boolean Functions with a NN

Logical Gates

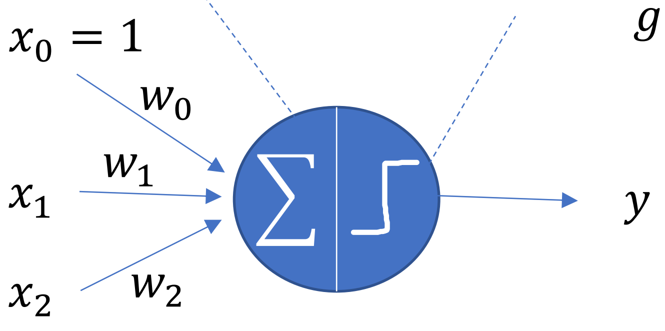
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

Source: https://medium.com/autonomous-agents/how-to-teach-logic-to-your-neuralnetworks-116215c71a49

Neural Network as a Logical AND Gate

Combination function: $z = w^T x = w_0 + w_1 x_1 + w_2 x_2$

Activation function $g(z) = \begin{cases} 0 \; ; if \; g(z) < 0.5 \\ 1 \; ; if \; g(z) \geq 0.5 \end{cases}$



Goal: find the value of weights which will enable the network to act as a particular gate.

AND Gate Truth Table

x_1	x_2	y: $g(x; w)$
0	0	0
0	1	0
1	0	0
1	1	1

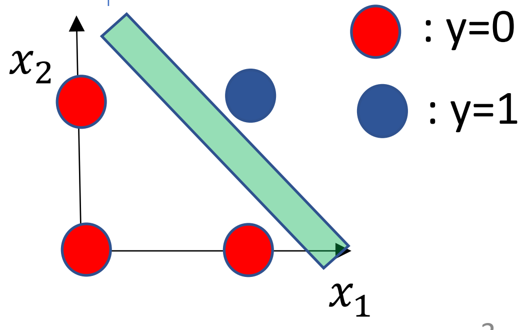
A 2-layered NN is called a Perceptron model.

Weights for the input layer

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 2 \end{bmatrix}$$

1) Case (0,0) : $w_0 + w_1 x_1 + w_2 x_2$ = w_0 = -3 ➔ y =0
2) Case (0,1) : $w_0 + w_2 x_2$ = -3 + (2)1 = -1 ➔ y=0
3) Case (1,0) : $w_0 + w_2 x_2$ = -3 + (2)1 = -1 ➔ y=0
4) Case (1,1) : $w_0 + w_1 x_1 + w_2 x_2$ = -3 + (2)1 +(2)1 = 1 ➔ y=1

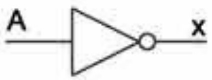
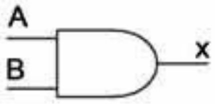
$-3 + 2x_1 + 2x_2 = 0$ ➔ $x_1 + x_2 = 3/2$



```
# Implementing AND Gate

class AND_Perceptron_model:

    def __init__(self, weights, threshold):
        self.weights=weights
        self.threshold = threshold

    def combination(self, x, w):
        return sum(x_i*w_i for x_i, w_i in zip(x, w))


    def stepactivation(self, sumproduct):
        print("Threshold: " , self.threshold)
        return 1.0 if sumproduct >= self.threshold else 0.0


    def fit(self, train):
        for row in train:
            layer_1 = row[:-1] # all attributes except the last column (y)
            print("For input ")
            print(layer_1)

            sumproduct= self.combination(layer_1, self.weights)
            layer_2= self.stepactivation(sumproduct )
            print("the sum is" ,sumproduct, "the output is", layer_2)

initial_w=[-3,2,2]
dataset = [[1,0,0,0],[1,0,1,0],[1,1,0,0],[1,1,1,1]]

model = AND_Perceptron_model (initial_w,0.5)
model.fit(dataset)

```

Emulating Boolean Functions with a NN

Logical Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

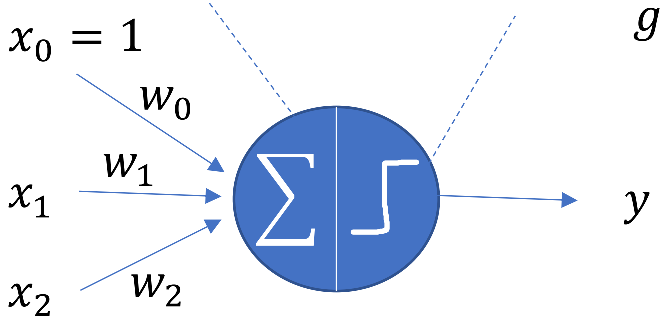
Source: https://medium.com/autonomous-agents/how-to-teach-logic-to-your-neuralnetworks-116215c71a49

Neural Network as a Logical NOR Gate

Combination function: $z = w^T x = w_0 + w_1 x_1 + w_2 x_2$

Activation function

$$g(z) = \begin{cases} 0 \; ; if \; g(z) < 0.5 \\ 1 \; ; if \; g(z) \geq 0.5 \end{cases}$$



Goal: find the value of weights which will enable the network to act as a particular gate.

NOR Gate Truth Table

x_1	x_2	y: $g(x; w)$
0	0	1
0	1	0
1	0	0
1	1	0

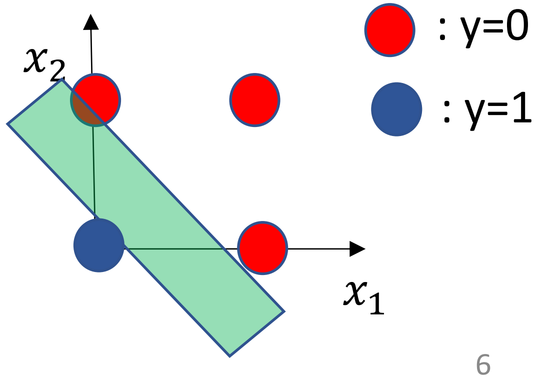
Weights of the input layer

$$\cancel{w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 2 \end{bmatrix}}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \\ -4 \end{bmatrix}$$

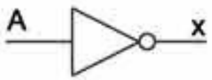
1) Case (0,0) : $w_0 + w_1 x_1 + w_2 x_2 = w_0$ = 3 ➔ y =1
2) Case (0,1) : $w_0 + w_2 x_2$ = 3 + (-4)1 = -1 ➔ y=0
3) Case (1,0) : $w_0 + w_2 x_2$ = 3 + (-4)1 = -1 ➔ y=0
4) Case (1,1) : $w_0 + w_1 x_1 + w_2 x_2$ = 3 + (-4)1 +(-4)1 = -5 ➔ y=0

$3 - 4x_1 - 4x_2$ = 0 ➔ $x_1 + x_2 = 3/4$



Emulating Boolean Functions with a NN

Logical Gates

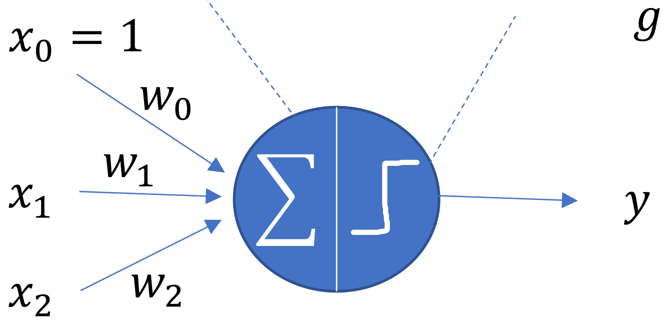
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

Source: https://medium.com/autonomous-agents/how-to-teach-logic-to-your-neuralnetworks-116215c71a49

Neural Network as a Logical XOR Gate

Combination function: $z = w^T x = w_0 + w_1 x_1 + w_2 x_2$

Activation function $g(z) = \begin{cases} 0 \;; if\; g(z) < 0.5 \\ 1 \;; if\; g(z) \geq 0.5 \end{cases}$



Goal: find the value of weights which will enable the network to act as a particular gate.

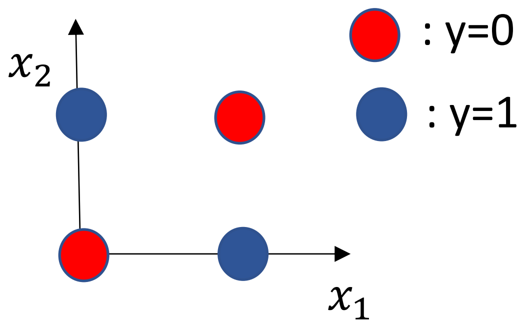
XOR Gate Truth Table

x_1	x_2	y: $g(x; w)$
0	0	0
0	1	1
1	0	1
1	1	0

Weights for the input layer

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \\ 1 \end{bmatrix}$$

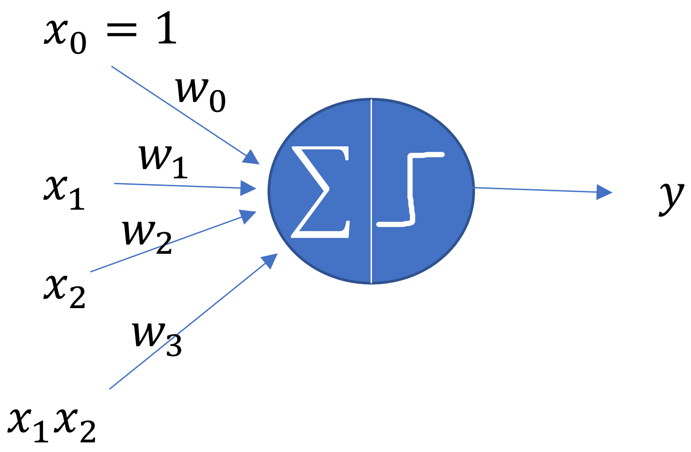
1) Case (0,0) : $w_0 + w_1 x_1 + w_2 x_2$ = w_0 = -0.5 ➔ y =0
2) Case (0,1) : $w_0 + w_2 x_2$ = -0.5 + (1)1 = 0.5 ➔ y=1
3) Case (1,0) : $w_0 + w_2 x_2$ = -0.5 + (1)1 = 0.5 ➔ y=1
4) Case (1,1) : $w_0 + w_1 x_1 + w_2 x_2$ = -0.5 + (1)1 +(1)1 = 1.5 ➔ y=1



Possible Solutions for Separability Problem

1. Add non-linear features called ”kernel trick”
2. Add extra layers known as “deep network”

Neural Network as a Logical XOR Gate: Kernel Trick

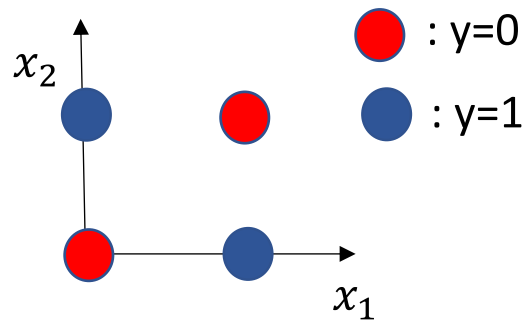


$$g(z) = \begin{cases} 0 \;; if\ g(z) < 0.5 \\ 1 \;; if\ g(z) \geq 0.5 \end{cases}$$

Goal: find the value of weights which will enable the network to act as a particular gate.

XOR Gate Truth Table

x_1	x_2	y: $g(x; w)$
0	0	0
0	1	1
1	0	1
1	1	0



Weights for the input layer

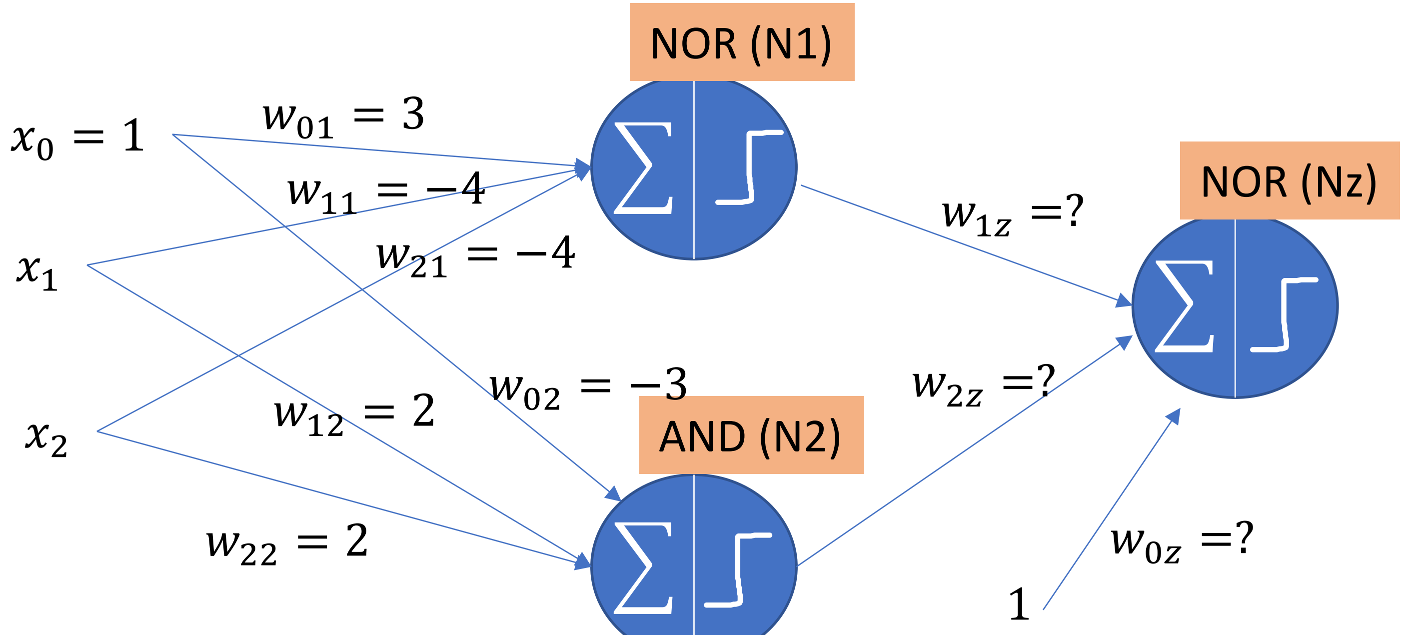
$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \\ 1 \\ -2 \end{bmatrix}$$

1) Case (0,0) : w_0 = -0.5 ➔ y =0
2) Case (0,1) : $w_0 + w_2 x_2$ = -0.5 + (1)1 = 0.5 ➔ y=1
3) Case (1,0) : $w_0 + w_2 x_2$ = -0.5 + (1)1 = 0.5 ➔ y=1
4) Case (1,1) : $w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1\, x_2$ = -0.5 + (1)1 +(1)1 + (-2)(1)(1)= -0.5 ➔ y=0

$$g(z) = -0.5 + x_1 + x_2 - 2\, x_1\, x_2$$

Neural Network as a Logical XOR Gate: Add Extra Layers

- XOR(x1,x2) can be thought of as NOR(NOR(x1,x2),AND(x1,x2))

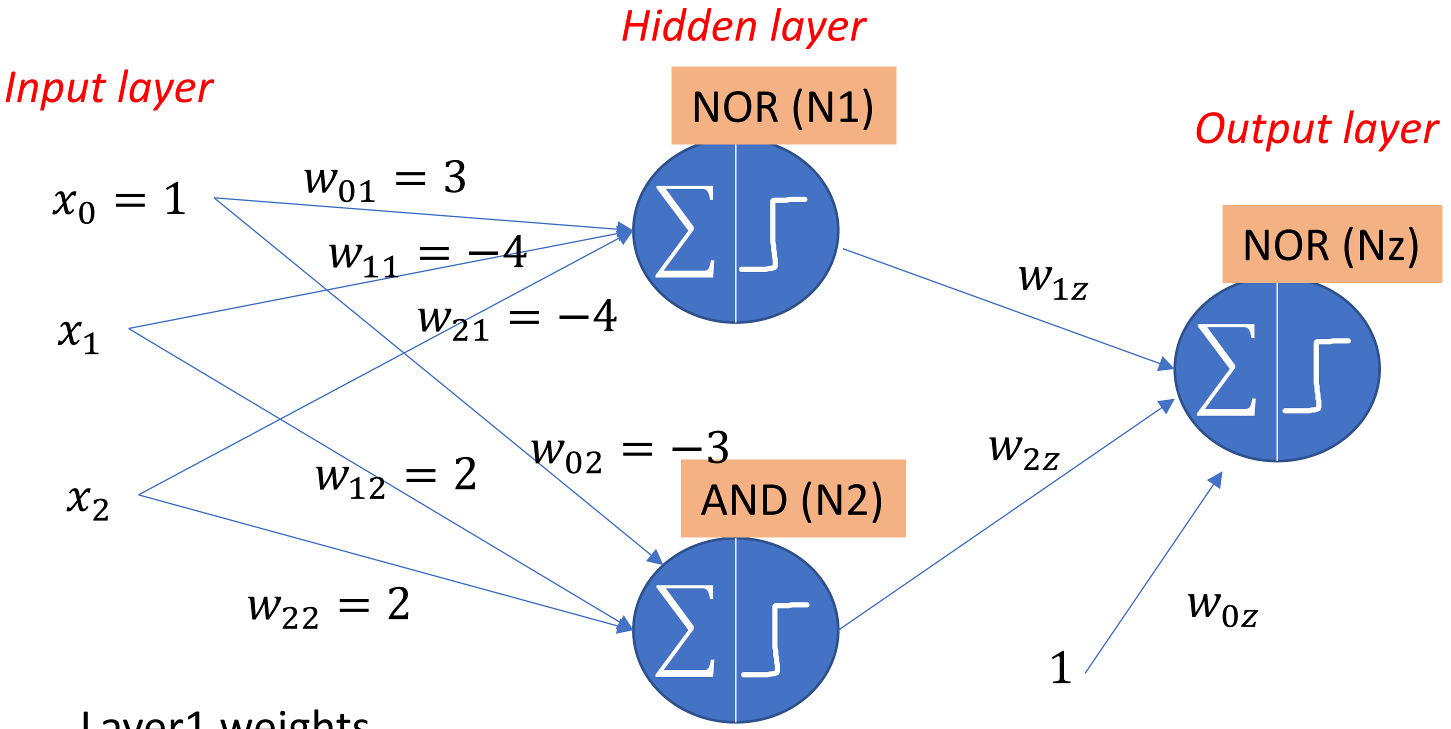


Truth Table for the network

x_1	x_2	$N1$	$N2$	Nz
0	0	?	?	0
0	1	?	?	1
1	0	?	?	1
1	1	?	?	0

3-Layer ANN Emulating XOR

$$g(z) = \begin{cases} 0 \;; if \;\; g(z) < 0.5 \\ 1 \;; if \;\; g(z) \geq 0.5 \end{cases}$$



Truth Table for the network

x_1	x_2	$N1$	$N2$	(XOR) Nz
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

Layer1 weights

$w_{01} = 3$
$w_{11} = -4$
$w_{21} = -4$
$w_{02} = -3$
$w_{12} = 2$
$w_{22} = 2$

Layer2 weights

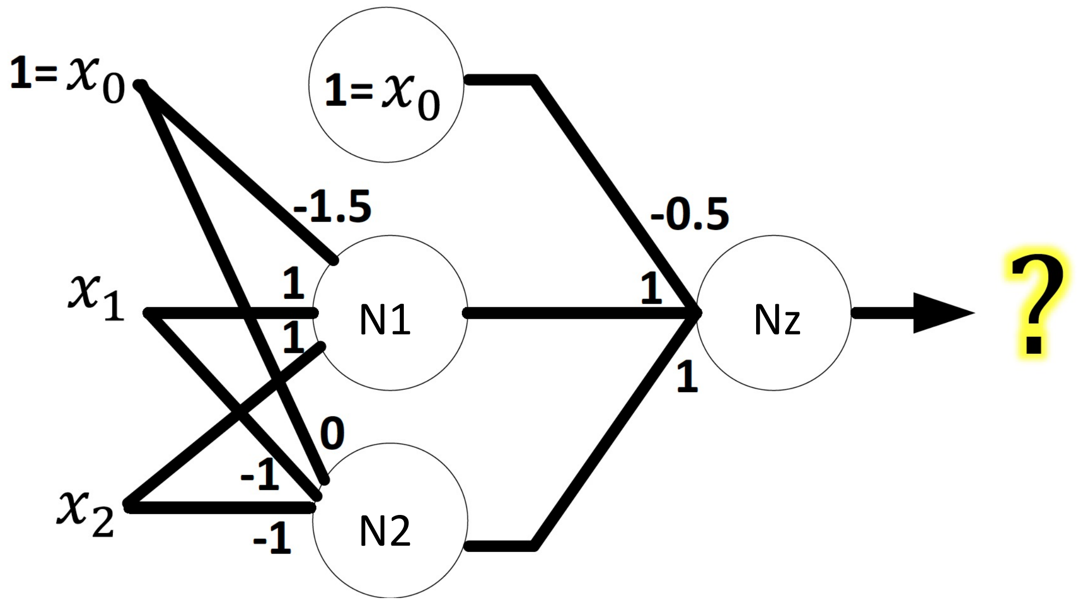
$w_{1z} = -2$
$w_{2z} = -2$
$w_{0z} = 1$

1) Case (1,0) :w_{0z} + w_{1z}= 1 + (-2) ➔ y =0
2) Case (0,0) : w_{0z} = 1 ➔ y=1
3) Case (0,0) : w_{0z} = 1 ➔ y=1
4) Case (0,1) : w_{0z} + w_{2z} = 1 + (-2) ➔ y=0

Practice Example

- What is this gate? Identify the logical gate for every neuron. Show all the steps taken to solve this. Assume the threshold for the activation function is 0.

x_1	x_2	y: $g(x;w)$
0	0	?
0	1	?
1	0	?
1	1	?



Optional Activity2

- Using the previous slide, design a similar question and include the solution.