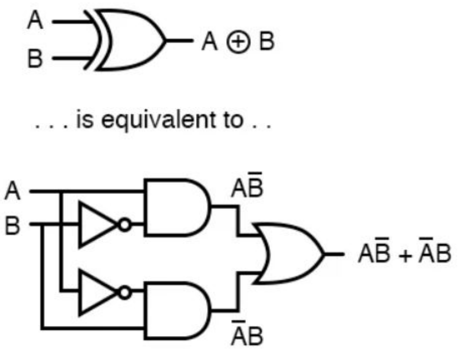
Designing XOR Gate

Submitted by: Imran Noor Saleh ; Student ID 19648
CS550 Machine Learning and Business Intelligence

Introduction

A digital logic gate that results in true (either 1 or HIGH) output when the number of true inputs is an odd count. An XOR gate implements an exclusive OR, i.e., a true output result if one, and only one, of the gate inputs, is true. The XOR gate problem was one of the early demonstrations of the power of neural networks and was used to show that neural networks could solve problems that traditional rule-based systems can't do.



Multilayer Presentation

```
      +------------------------------------------------------------+
      |      "NAnd"   +---------+                 "XOr" Function   |
      | 1.0-->2.0--->|         |                                  |
      |              |         |                                  |
X --->|----> -0.75-->| Neuron  |-->Z2--                           |
      |\             |         |      \                           |
Y---->|-\--> -0.75-->|         |       \ "And"   +---------+      |
      |\ \           +---------+        ->0.75-->|         |      |
    |  \ \                                      | Neuron  |-->Z3-->|-->Z
      |  \ \  "Or"   +---------+       -->0.75-->|         |      |
      |   \ -->1.5-->|         |       /         +---------+      |
      |    \         | Neuron  |-->Z1--                           |
      |     -->1.5-->|         |                                  |
      |              +---------+                                  |
      +------------------------------------------------------------+
```

With Bias

Implementation Contd.

Z := (W1 * X + W2 * Y >= T)

where T := 1.0.

Desired "**And**" Function

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Implementation Contd.

Loop 1:

W1=W2=0

Z := (0 * X + 0 * Y >= T)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	0

Implementation Contd.

Loop 2:

W1=W2=0.5

Z := (0.5 * X + 0.5 * Y >= T)

X	**Y**	**Z**
0	0	0
0	1	0
1	0	0
1	1	1

Implementation Contd.

Design **OR** Gate to calculate the values of **W1,W2 and Y**

Z := (W1 * X + W2 * Y >= T)

where T := 1.0.

Desired "**OR**" Function

X	**Y**	**Z**
0	0	0
0	1	1
1	0	1
1	1	1

Implementation Contd.

Loop 1:

W1=W2=0

Z := (0* X + 0 * Y >= T)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	0

Implementation Contd.

Loop 2:

W1=W2=0.5

Z := (0.5* X + 0.5 * Y >= T)

X	**Y**	**Z**
0	0	0
0	1	0
1	0	0
1	1	1

Implementation Contd.

Loop 3:

W1=W2=1.0

Z := (1.0* X + 1.0 * Y >= T)

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Implementation Contd.

Design **NAND** Gate to calculate the values of **W1,W2 and Y**

Z := (W0 * C + W1 * X + W2 * Y >= T)

where T := 1.0.

The bias C for NAND is 1.0

Desired **"NAND"** Function

X	**Y**	**Z**
0	0	1
0	1	1
1	0	1
1	1	0

Implementation Contd.

Loop 1 :

W0=0.0

W1=W2=0.5

Z := (0 * 1.0 + 0.5 * X + 0.5 * Y >= T)

C	X	Y	Z
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Implementation Contd.

Loop 2:

W0=0.5

W1=W2=0.5

Z := (0.5* 1.0 + 0.5 * X + 0.5 * Y >= T)

C	X	Y	Z
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Implementation Contd.

Loop 3:

W0=1.0

W1=W2=0.5

Z := (1.0* 1.0 + 0.5 * X + 0.5 * Y >= T)

C	X	Y	Z
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Implementation Contd.

Loop 4:

W0=1.0

W1=W2=0.0

Z := (1.0* 1.0 + 0.0 * X + 0.0 * Y >= T)

C	X	Y	Z
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Implementation Contd.

Loop 5:

W0=1.0

W1=W2=-0.5

Z := (1.0* 1.0 + -0.5 * X + -0.5 * Y >= T)

C	X	Y	Z
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Implementation Contd.

Loop 6:

W0=1.5

W1=W2=-0.5

Z := (1.5* 1.0 + -0.5 * X + -0.5 * Y >= T)

C	X	Y	Z
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Test

- What is the formula for

Z1 := X "AND" Y

Z1 := (0.5 * X + 0.5 * Y >= 1.0)

- **What is the formula for**

Z1 := X "OR" Y

Z1 := (1.0 * X + 1.0 * Y >= 1.0)

Test

- **What is the formula for**

Z1 := X "NAND" Y

Bias is +1.5 , C = 1; W0 = 1.5; W1=W2 = -0.5

Z2 := (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0)

Z2 := (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0)

Z2 := (- 0.5 * Y >= 0.5 * X + -1.5 * 1.0 + 1.0)

Z2 := (-0.5 * Y >= 0.5 * X – 0.5)

Z2 := (Y <= -X + 1.0)

Test

- **What is the formula for**

Z1 := X "Or" Y

Z2 := X "NAND" Y

Z := Z3 := Z1 "AND" Z2

Z := (X "OR" Y) "AND" (X "NAND" Y)

Z := (1.0 * X + 1.0 * Y >= 1.0) "AND" (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0)

Z := (0.5 * (1.0 * X + 1.0 * Y >= 1.0) + 0.5 * (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0) >= 1.0)

Z := (0.5 * (1.0 * X + 1.0 * Y >= 1.0) + 0.5 * (1.5 + -0.5 * X + -0.5 * Y >= 1.0) >= 1.0)

Test

- **Step 4: Please prove that your designed XOR Gate work**
  - **X=1, Y=1**
  - **X=1, Y=0**
  - **X=0, Y=1**
  - **X=0, Y=0**

Z1 := X "Or" Y

Z2 := X "NAND" Y

Z := Z3 := Z1 "AND" Z2

Z := (X "OR" Y) "AND" (X "NAND" Y)

Z := (1.0 * X + 1.0 * Y >= 1.0) "AND" (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0)

Z := (0.5 * (1.0 * X + 1.0 * Y >= 1.0) + 0.5 * (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0) >= 1.0)

Z := (0.5 * (1.0 * X + 1.0 * Y >= 1.0) + 0.5 * (1.5 + -0.5 * X + -0.5 * Y >= 1.0) >= 1.0)

Test

Take X=1 ,Y=1

Z := (0.5 * (1.0 * 1.0 + 1.0 * 1.0 >= 1.0) + 0.5 * (1.5 + -0.5 * 1.0 + -0.5 * 1.0 >= 1.0) >= 1.0)

Z := (0.5 * (1.0 + 1.0 >= 1.0) + 0.5 * (1.5 + -0.5 + -0.5 >= 1.0) >= 1.0)

Z := (0.5 * (2.0 >= 1.0) + 0.5 * (0.5 >= 1.0) >= 1.0)

Z := (0.5 * (true) + 0.5 * (false) >= 1.0)

Z := (0.5 * 1 + 0.5 * 0 >= 1.0)

Z := (0.5 + 0.0 >= 1.0)

Z := (false)

Z := 0

Test

Take X=1 ,Y=0

Z := (0.5 * (1.0 * 1.0 + 1.0 * 0.0 >= 1.0) + 0.5 * (1.5 + -0.5 * 1.0 + -0.5 * 0.0 >= 1.0) >= 1.0)

Z := (0.5 * (1.0 + 0.0 >= 1.0) + 0.5 * (1.5 + -0.5 + -0.0 >= 1.0) >= 1.0)

Z := (0.5 * (1.0 >= 1.0) + 0.5 * (1.0 >= 1.0) >= 1.0)

Z := (0.5 * (true) + 0.5 * (true) >= 1.0)

Z := (0.5 * 1 + 0.5 * 1 >= 1.0)

Z := (0.5 + 0.5 >= 1.0)

Z := (true)

Z = 1

Test

Take X=1 ,Y=0

Z := (0.5 * (1.0 * 1.0 + 1.0 * 0.0 >= 1.0) + 0.5 * (1.5 + -0.5 * 1.0 + -0.5 * 0.0 >= 1.0) >= 1.0)

Z := (0.5 * (1.0 + 0.0 >= 1.0) + 0.5 * (1.5 + -0.5 + -0.0 >= 1.0) >= 1.0)

Z := (0.5 * (1.0 >= 1.0) + 0.5 * (1.0 >= 1.0) >= 1.0)

Z := (0.5 * (true) + 0.5 * (true) >= 1.0)

Z := (0.5 * 1 + 0.5 * 1 >= 1.0)

Z := (0.5 + 0.5 >= 1.0)

Z := (true)

Z = 1

Test

Take X=0 ,Y=0

Z := (0.5 * (1.0 * 0.0 + 1.0 * 0.0 >= 1.0) + 0.5 * (1.5 + -0.5 * 0.0 + -0.5 * 0.0 >= 1.0) >= 1.0)

Z := (0.5 * (0.0 + 0.0 >= 1.0) + 0.5 * (1.5 + -0.0 + -0.0 >= 1.0) >= 1.0)

Z := (0.5 * (0.0 >= 1.0) + 0.5 * (1.5 >= 1.0) >= 1.0)

Z := (0.5 * (false) + 0.5 * (true) >= 1.0)

Z := (0.5 * 0 + 0.5 * 1 >= 1.0) Z := (0.0 + 0.5 >= 1.0)

Z := (false)

Z = 0

Test

OR		**NAND**		**XOR**
X Y | Z1		X Y | Z2		X Y | Z3
----------		----------		----------
0 0 | 0		0 0 | 1		0 0 | 0
0 1 | 1	**AND**	0 1 | 1	==>	0 1 | 1
1 0 | 1		1 0 | 1		1 0 | 1
1 1 | 1		1 1 | 0		1 1 | 0

From Above Calculations, Hence **“OR”** AND “**NAND**” GATE Operations Output is **XOR** GATE.