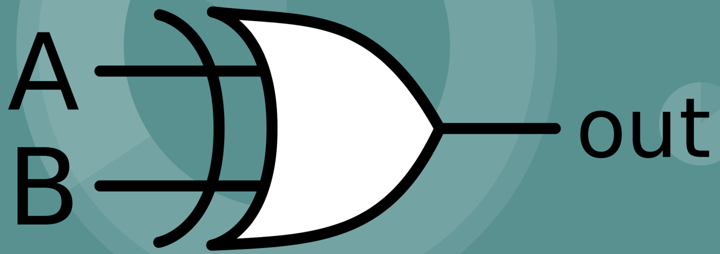
CS550 - Machine Learning and Business Intelligence



Project :Design XOR Gate

By: Maranata Muluneh

Instructor: Dr. Chang, Henry

2023

Table of Content

- Introduction
- Theory
- Implementation
- Conclusion
- References

Introduction

An **XOR gate** (Exclusive OR gate) is a logical gate that performs an exclusive OR operation on two input signals. It produces an output of 1 only when the two input signals are different. In other words, if either one input is high (1) and the other is low (0), the XOR gate will output 1, otherwise it will output 0.

Theory

The truth table for an XOR gate with inputs A and B and output Y can be represented as follows:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

In digital electronics, XOR gates are used extensively in circuit design, particularly for data processing and error detection. They are also used in encryption algorithms and communication protocols.

Theory

A **neural network** is a type of machine learning model inspired by the structure and function of the human brain. It consists of a large number of interconnected processing nodes or neurons that work together to solve a specific problem.

```
          +-----------------------------+
          |                             |
          |               +---------+   |
          |  C --->W0--->|         |   |
          |               |         |   |
X  --->|------->W1--->| Neuron  |--->|---> Z
          |               |         |   |
Y---->|------->W2--->|         |   |
          |               +---------+   |
          |                             |
          +-----------------------------+
```

Note:

- Weights: W0, W1, W2
- Bias: W0 * C
  Where C is a constant
- Inputs: X, Y
- Transfer Function (Activation Function):

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

Theory

An **activation function** is a mathematical function applied to the output of each neuron in a neural network. It introduces non-linearity into the network, allowing it to model complex, non-linear relationships between inputs and outputs.

```
      +--------------------------------------------+
      |                  +-----+------------+       |
      |  C --->W0--->|     |            |       |
      |                  |     | Activation |       |
X --->|-------->W1--->| Sum | Function   |--->|---> Z
      |                  |     |            |       |
Y---->|-------->W2--->|     |            |       |
      |                  +-----+------------+       |
      +--------------------------------------------+
```

Note:

- Sum = W0 * C + W1 * X + W2 * Y
- For Step activation function

```
Z := ( Sum >= T )

Note:
    if Sum >= T
    then Z = 1
```

Theory

How to train the gates?

- Forward Pass (Feedforward) training algorithm

```
Z := ( W0 * C + W1 * X + W2 * Y >= T )
     where T := 1.0

if ( W0 * C + W1 * X + W2 * Y >= T )
then output Z is 1
else output Z = 0
```

Theory

How to train the gates?

- Set the input (X,Y) to some possible value such as (0,0), (0,1), (1,0), or (1,1).

Forward process

- Calculate the output Z for the given input (X,Y).

Backward process

- Adjust weight
- If the output Z is too low, increase the weights which had inputs that were "1".
- If the output Z is too high, decrease the weights which had inputs that were "1".
- Continue looping through this process until each possible input combination gives the right

Implementation

Project: Design XOR Gate

```
      +-----------------------------------------------------------+
      |      "NAND"  +--------+                    "XOR" Function |
      | 1.0-->1.0--->|        |                                   |
      |              |        |                                   |
X --->|----> ???? -->| Neuron |-->Z2--                            |
      |\             |        |       \                           |
Y---->|-\--> ???? -->|        |        \ "AND"   +--------+       |
      |\ \           +--------+         ->????-->|        |       |
      | \ \                                      | Neuron |-->Z3-->|-->Z
      |  \ \  "Or"    +--------+       -->????-->|        |       |
      |   \ -->???-->|        |       /          +--------+       |
      |    \         | Neuron |-->Z1--                            |
      |     -->???-->|        |                                   |
      |              +--------+                                   |
      +-----------------------------------------------------------+
```

Implementation

Step 1: Using the following rules to design your own AND Gate, OR Gate, and NAND Gate

Using the forward/backward process

- Forward process
  Calculate the output Z for the given input (X,Y).
- Backward process
  Adjust weights
  + If the output Z is too low, increase the weights by 0.5
    which had inputs that were "1".
  + If the output Z is too high, decrease the weights by 0.5

Using the step activation function

Z := (W1 * X + W2 * Y >= T)
  where T := 1.0

Implementation

Project: Design XOR Gate

1. AND

Desired
"And"
Function

X Y | Z
----
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1

Loop 1
W1=W2=0
Function

X Y | Z
----
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 0

Loop 2
W1=W2=0.5
Function

X Y | Z
----
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1

We have the desired result here on loop 1

Hence the formula will be:

Z := (0.5 * X + 0.5 * Y >= 1)

Implementation

Project: Design XOR Gate

2. OR

Desired
"OR"
Function

X Y | Z
----
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1

Loop 1
W1=W2=0
Function

X Y | Z
----
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 0

Loop 2
W1=W2=0.5
Function

X Y | Z
----
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1

Loop 3
W1=W2=1.0
Function

X Y | Z
----
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1

We have the desired result here on loop 3

Hence the formula will be:

Z := (1.0 * X + 1.0 * Y >= 1)

Implementation

Project: Design XOR Gate

3. NAND

Using the step activation function

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

where T := 1.0

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

then output is 1

else output = 0

The bias C for NAND is 1.0

Desired "NAND" Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 0

Loop 1
W0 =0.0
W1=W2=0.5
Function

C X Y | Z
----
1 0 0 | 0
1 0 1 | 0
1 1 0 | 0
1 1 1 | 1

Loop 2
W0 =0.5
W1=W2=0.5
Function

C X Y | Z
----
1 0 0 | 0
1 0 1 | 1
1 1 0 | 1
1 1 1 | 1

Loop 3
W0 =1.0
W1=W2=0.5
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 1

Implementation

Project: Design XOR Gate

Loop 4
W0 =1.0
W1=W2=0.0
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 1

Loop 5
W0 =1.0
W1=-0.5, W2=0.0
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 0
1 1 1 | 0

Loop 6
W0 =1.0
W1=0.0, W2=0.0
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 1

Loop 7
W0 =1.0
W1=0.0, W2=-0.5
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 0
1 1 0 | 1
1 1 1 | 0

Loop 8
W0 =1.5
W1=0.0, W2=-0.5
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 1

Loop 9
W0 =1.5
W1=-0.5, W2=-0.5
Function

C X Y | Z
----
1 0 0 | 1
1 0 1 | 1
1 1 0 | 1
1 1 1 | 0

Z := (1.5 * C -0.5 * X -0.5* Y >= 1)

Implementation

Step 2: What is the formula for

"AND"

Z := (0.5 * X + 0.5 * Y >= 1)

"OR"

Z := (1.0 * X + 1.0 * Y >= 1)

"NAND"

Z := (1.5 * C -0.5 * X -0.5* Y >= 1)

Implementation

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

Test 1: X=1, Y=1	Desired Result
Z1: = (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0) = 1.5-0.5-0.5 := 0 Z2: = (1* X + 1* Y >= 1.0) = 1 + 1 := 1 Z3: = (0.5* Z1 + 0.5* Z2 >= 1.0) = 0.5*0 + 0.5*1 := 0	XOR ----------- X Y | Z ----------- 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Implementation

Test 2: X=1, Y=0	Desired Result
Z1: = (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0) = 1.5-0.5*1-0.5*0 := 1 Z2: = (1* X + 1* Y >= 1.0) = 1*1 + 1*0 := 1 Z3: = (0.5* Z1 + 0.5* Z2 >= 1.0) = 0.5*1 + 0.5*1 := 1	XOR ----------- X Y | Z ----------- 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Implementation

Test 3: X=0, Y=1	Desired Result
Z1: = (1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0) = 1.5*1.0-0.5*0-0.5*1 := 1 Z2: = (1* X + 1* Y >= 1.0) = 1*0 + 1*1 := 1 Z3: = (0.5* Z1 + 0.5* Z2 >= 1.0) = 0.5*1 + 0.5*1 := 1	XOR ----------- X Y | Z ----------- 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Implementation

```
Z1: = ( 1.5 * 1.0 + -0.5 * X + -0.5 * Y >= 1.0
)

      = 1.5-0.5*0-0.5*0  := 1


Z2: = (1* X + 1* Y >= 1.0 )

      = 1*0 + 1*0  := 0

Z3: = (0.5* Z1 +  0.5* Z2 >= 1.0 )

      = 0.5*1 + 0.5*0 := 0
```

XOR

X Y | Z

----

0 0 | 0

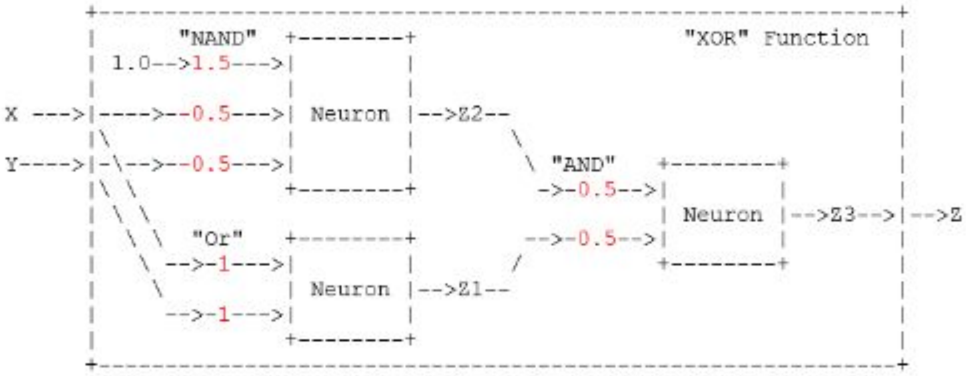
0 1 | 1

1 0 | 1

1 1 | 0

Implementation

Conclusion:



Conclusion

Overall, The projects aims to design XOR gate using the Neural Network, and implementing Forward and Backward Pass Algorithm. By doing so we have the basic steps by which we implement the Forward and Backward process with the activation function.

Besides using Neural network to design Logic gateway, we have different application including image and speech recognition, natural language processing, and robotics, among others.

References

- What are neural networks? IBM. (n.d.). Retrieved April 3, 2023, from https://www.ibm.com/topics/neural-networks

- M, D. (2022, June 29). XOR problem with neural networks: An explanation for Beginners. Analytics India Magazine. Retrieved April 3, 2023, from https://analyticsindiamag.com/xor-problem-with-neural-networks-an-explanation-for-beginners/#:~:text=The%20XOR%20problem%20with%20neural%20networks%20can%20be%20solved%20by,the%20XOR%20logic%20gets%20executed.