Project: Design XOR Gate

By Akanksha Singh, 19745

Introduction

In this project, we will make use of the model of neural network which includes

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

We will train gates using the forward pass raining algorithm for the following formula:

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

Table of Contents

Step 1: General Idea of Designing a XOR Gate

The truth-table for the "XOR" function.

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

Example: XOR function created using a Neural Network

```
      +------------------------------------------------------------+
      |      "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-->|        |                                    |
      |              +--------+                                    |
      +------------------------------------------------------------+
```

Our neural network equation can be created by combining neural equations.

```
Z1 := X  "Or"   Y
Z2 := X  "NAND" Y
Z := Z3 := Z1 "AND" Z2
Z := ( X "Or" Y ) "AND" ( X "NAND" Y )
Z := ( 1.5 * X + 1.5 * Y >= 1.0 ) "AND"
     ( 2.0 + -0.75 * X + -0.75 * Y >= 1.0 )
Z := ( 0.75 * ( 1.5 * X + 1.5 * Y >= 1.0 ) +
       0.75 * ( 2.0 + -0.75 * X + -0.75 * Y >= 1.0 ) >= 1.0 )
```

Step 2: Design your own AND, OR, NAND Gates

OR Gate Desired Truth Table:

```
-----------
X  Y  |  Z1
-----------
0  0  |  0
0  1  |  1
1  0  |  1
1  1  |  1
```

If W1=W2=1 then using **Step Activation Function**, Z1 = (1*X)+(1*Y) >=T where T=1.

Step 2: Design your own AND, OR, NAND Gates

NAND Gate Desired Truth Table with Bias as 1:

C X Y | Z2

----

1 0 0 | 1

1 0 1 | 1

1 1 0 | 1

1 1 1 | 0

If W0=1.5, W1=W2=-0.5 then using **Step Activation Function**, Z2 = (1.5*1)+(-0.5*X)+(-0.5*Y) >=T where T=1.

Combining Neural Equations

Z1 := X "Or" Y

Z2 := X "NAND" Y

Z := Z3 := Z1 "AND" Z2

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

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

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

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

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

Note: for AND

```
  AND
-----------
X  Y  |  Z
-----------
0  0  |  0
0  1  |  0
1  0  |  0
1  1  |  1
```

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

Step 3: Formulas for AND, OR, NAND

- **Z1 := X "OR" Y :**

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

- **Z2 := X "NAND" Y**

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

- **Z := Z3 := Z1 "AND" Z2**

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

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

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

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

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

Step 4: Proof that Designed XOR Gate works

For, X=1, Y=1 :

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

0.5 * (1.5*1 + -0.5 * 1 + -0.5 * 1 >= 1.0) >= 1.0)

Z := (0.5 * (2>= 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>= 1.0)

Z := (**False**)

Step 4: Proof that Designed XOR Gate works

For, X=1, Y=0 :

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

0.5 * (1.5 + -0.5 * 1 + -0.5 *0 >= 1.0) >= 1.0)

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

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

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

Z := (1>= 1.0)

Z := (**TRUE**)

Step 4: Proof that Designed XOR Gate works

For, X=0, Y=1 :

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

0.5 * (1.5 + -0.5 * 0 + -0.5 *1 >= 1.0) >= 1.0)

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

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

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

Z := (1>= 1.0)

Z := (**TRUE**)

Step 4: Proof that Designed XOR Gate works

For, X=0, Y=0 :

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

0.5 * (1.5 + -0.5 * 0 + -0.5 *0 >= 1.0) >= 1.0)

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

Z := (**False**)

Conclusion

We can conclude that the XOR gate designed by us satisfied all the values of X and Y for the XOR Gate Table. The desired Truth Table was achieved as below with the calculated formula:

Z := (0.5 * (1 * X + 1 * Y >= 1.0) + 0.5 * (1.5*C + -0.5 * X + -0.5 *Y >= 1.0) >= 1.0)

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

Step 5,6: Github Link

https://github.com/codeyogg/Machine_learning/tree/main/Design_XOR_Gate