Design XOR Gate

Fatema Nagori

Table of Content

Introduction

Design

Implementation

Test

Conclusion

References

Introduction

Our black box for the "XOR" function now has three neurons in it. A collection of neurons connected together is a "network" of neurons. Thus, the "XOR" function has been created using a "neural network".

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

Design

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
```

Implementation

Step 2: Using the following rules to design your own AND Gate, OR Gate, and NAND Gate

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

    which had inputs that were "1".

Using 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 Gates is 1.0

Implementation

Step 3: Solve for Z1 := X "AND" Y

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

Desired "AND" Function				Loop 1 W0=0.0 W1=W2=0.5 Function			
C	X	Y	Z	C	X	Y	Z
1	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1

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

==> Z := (Y >= -1.0 * X + 2.0)

Solve for Z1 := X "OR" Y

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

```
                Loop 1          Loop 2
Desired         W0=0.0          W0=0
"OR"            W1=W2=0.5       W1=W2=1
Function        Function        Function

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

Z := (0.0 * 1 + 1 * X + 1 * Y >= 1.0)

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

Solve for Z1 := X "NAND" Y

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

where T := 1.0.

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

==> Z := (Y >= -1.0 + X)

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
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

Loop 4
W0=1
W1=0,W2=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
W1=-0.5,W2=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
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 7
W0=1
W1=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,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

TEST

Step 4: Please prove that your designed XOR Gate work

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

Z := (0.5 >= 1.0)

Z := (False)

Z := 0

TEST

- **X=1, Y=0**

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

Z := (True)

Z := 1

TEST

- **X=0, Y=1**

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

Z := (True)

Z := 1

TEST

- **X=0, Y=0**

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

Z := 0

CONCLUSION:

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

OR

X	Y	Z1
0	0	0
0	1	1
1	0	1
1	1	1

AND

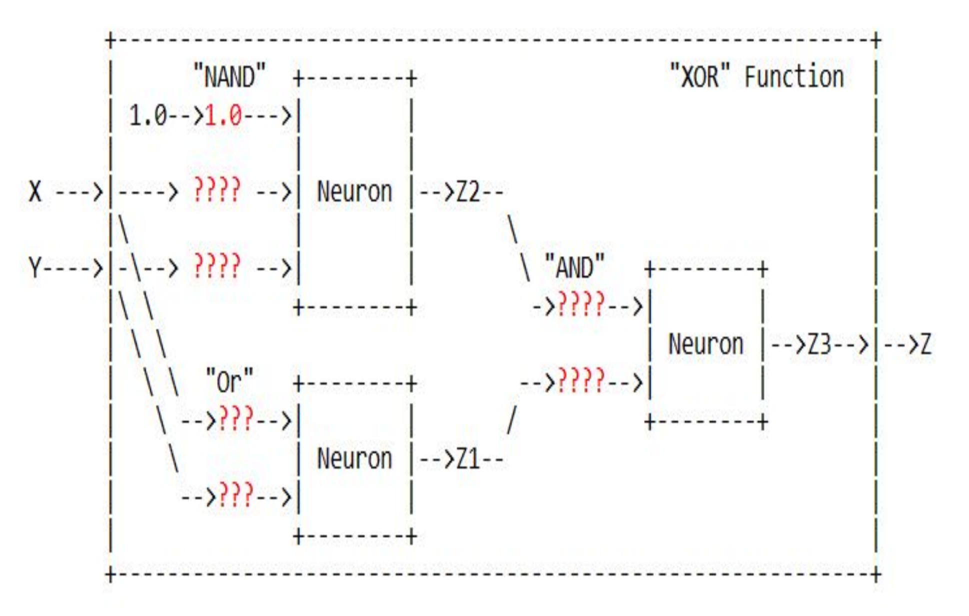
NAND

X	Y	Z2
0	0	1
0	1	1
1	0	1
1	1	0

=

XOR

X	Y	Z3
0	0	0
0	1	1
1	0	1
1	1	0



REFERENCES

PROF Chang's Material

THANK YOU