Week 10 HW1
Design XOR Gate

Khoi Duong
Prof. Chang
CS550

Understanding the problem

Please refer to A Neural Network Primer to solve this question.

Step 1: Study the general idea on how to design XOR Gate

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

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

What we know

- 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 ouput is 1
else output = 0
```

- The bias C for NAND is 1.0

Formula for Z1 := X "AND" Y

Thus, we have W1 = W2 = 0.5.
The formula is Z1:= (0.5X + 0.5Y >= T)
where T := 1.0

```
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
###########################
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
```

Formula for Z1 := X "OR" Y

Thus, we have W1 = W2 = 1.
The formula is Z1:= (X + Y >= T)
where T := 1.0

```
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
############################
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
```

```
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
Function

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

Formula for Z2 := X "NAND" Y

```
NAND := NOT AND
==> Z := !( 0.5 * X + 0.5 * Y >= 1.0 )
==> Z := ( 0.5 * X + 0.5 * Y < 1.0 )
==> Z := ( -0.5 * X - 0.5 * Y > -1.0 )
==> Z := (+1.0 * 1.0 + -0.5 * X - 0.5 * Y > +1.0 * 1.0 -1.0 )
==> Z := (+1.0 * 1.0 + -0.5 * X + -0.5 * Y > 0 )
```

The formula is Z2:= (+1.0 * 1.0 - 0.5 * X - 0.5 * Y > T)
where T := 0

The new schema becomes

```
      +------------------------------------------------------------+
      |      "NAND"  +--------+                    "XOR" Function  |
      | 1.0-->1.0--->|        |                                    |
      |              |        |                                    |
X --->|----> -0.5--->| Neuron |-->Z2--                             |
      |\             |        |       \                            |
Y---->|-\--> -0.5--->|        |        \ "AND"   +--------+        |
      |\ \           +--------+         ->0.5--->|        |        |
      | \ \                                      | Neuron |-->Z3-->|-->Z
      |  \ \  "Or"   +--------+        -->0.5--->|        |        |
      |   \ -->1.0-->|        |       /          +--------+        |
      |    \         | Neuron |-->Z1--                             |
      |     -->1.0-->|        |                                    |
      |              +--------+                                    |
      +------------------------------------------------------------+
```

Formula for Z := Z3 := Z1 "AND" Z2

```
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"
     ( 1.0 + -0.5 * X + -0.5 * Y > 0 )
Z := ( 0.5 * ( 1.0 * X + 1.0 * Y >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 * X + -0.5 * Y > 0 ) >= 1.0 )
```

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

We have the formula above. Next, we have to prove that the designed XOR gate work.

Test case a. X = 1, Y = 1

```
a. X = 1, Y = 1
Z := ( 0.5 * ( 1.0 * 1 + 1.0 * 1 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 * 1 + -0.5 * 1 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 1.0 + 1.0 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 + -0.5 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 2.0 >= 1.0 ) +
       0.5 * ( 0.0 >  0.0 ) >= 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 case b. X = 1, Y = 0

```
b. X = 1, Y = 0
Z := ( 0.5 * ( 1.0 * 1 + 1.0 * 0 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 * 1 + -0.5 * 0 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 1.0 + 0.0 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 + 0.0 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 1.0 >= 1.0 ) +
       0.5 * ( 0.5 >  0.0 ) >= 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 case c. X = 0, Y = 1

```
c. X = 0, Y = 1
Z := ( 0.5 * ( 1.0 * 0 + 1.0 * 1 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 * 0 + -0.5 * 1 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 0.0 + 1.0 >= 1.0 ) +
       0.5 * ( 1.0 + 0.0 + -0.5 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 1.0 >= 1.0 ) +
       0.5 * ( 0.5 >  0.0 ) >= 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 case d. X = 0, Y = 0

```
d. X = 0, Y = 0
Z := ( 0.5 * ( 1.0 * 0 + 1.0 * 0 >= 1.0 ) +
       0.5 * ( 1.0 + -0.5 * 0 + -0.5 * 0 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 0.0 + 0.0 >= 1.0 ) +
       0.5 * ( 1.0 + 0.0 + 0.0 > 0 ) >= 1.0 )
Z := ( 0.5 * ( 0.0 >= 1.0 ) +
       0.5 * ( 1.0 >  0.0 ) >= 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
```

Conclusion

All cases satisfy the desire “XOR” function. Therefore, the design XOR gate works perfectly.

Reference

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/deep_learning/slide/exercise_deep_learning.html#xor

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/neural_network/slide/ann.html

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/deep_learning/slide/exercise_deep_learning.html#create_nn