From: https://medium.com/swlh/learning-a-xor-function-with-feedforward-neural-networks-74ba3841f1c0

Finding the XOR weights:

First take the regression equation:

$\hat{y} = w^T x + b$ — eq-1

And the inputs: $\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ and their corresponding outputs: $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Plugging the values to eq-1 gives:

First input $[0 \quad 0] -> 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} w^T + b -> b = 0$

Second input $[0 \quad 1] -> 1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} w^T + 0 -> w_2 = 1$

Third input $[1 \quad 0] -> 1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} w^T + 0 -> w_1 = 1$

For the final output we have to keep b to see the issue with linear regression on a nonlinear function.

Final input $[1 \quad 1] -> 0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} w^T + b -> b + 2 = 0$; Now taking $b = 0$ makes the equation invalid. This shows how the linear regression cannot approximate a nonlinear function such as XOR. Therefore lets take the final input and give it to a simple neural network model

$xW = [a \quad b] \begin{bmatrix} x & z \\ y & w \end{bmatrix} = \begin{bmatrix} ax + by & az + bw \end{bmatrix}$ -eq—2

Since we know $w_1 = 1, w_2 = 1$ works for the first Three inputs, We can replicate and set it here $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

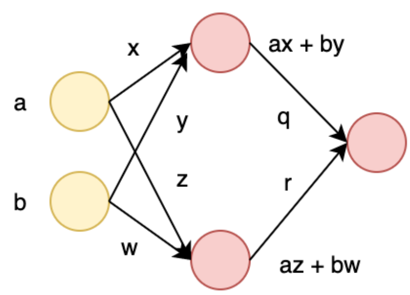
From eq—2

$[1 \quad 1] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = [2 \quad 2]$. Then the rest of the inputs.

$[0 \quad 0] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = [0 \quad 0]$

$[0 \quad 1] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = [1 \quad 1]$

$[1 \quad 0] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = [1 \quad 1]$



So we have now the input mapping in the representation space (The hidden layer)

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} -> \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

Now it is clear that the first 3 rows of the representation space outputs has the expected pattern of the output space $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$. So the idea is to find a vector that would:

1. Keep the first column as it is (Since the fist 3 elements are the expected output)
2. Adjust the last column so that we can find a pair of output weights to map the 2 —> 0 and keep the rest of the elements as they are.

Take $c = [0 \quad -1]$ and extend now the equation 2 with the new bias term c

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} - [0 \quad -1] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Now we must handle the -1 case (Otherwise the output mapping would not be easy) by using a ReLU layer.

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \text{—ReLU—>} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Finally we can select a pair of weights to handle the last row of the representation space as follows

$$[2 \quad 1] \begin{bmatrix} q \\ r \end{bmatrix} = 0$$

Which gives $q = 1, r = -2$. Also these weights map the first three elements to their corresponding outputs.

For the rest of the representation space

$$[0 \quad 0] \begin{bmatrix} 1 \\ -2 \end{bmatrix} = 0$$

$$[1 \quad 0] \begin{bmatrix} 1 \\ -2 \end{bmatrix} = 1$$

$$[1 \quad 0] \begin{bmatrix} 1 \\ -2 \end{bmatrix} = 1$$

———————————————— END ————————————————————————