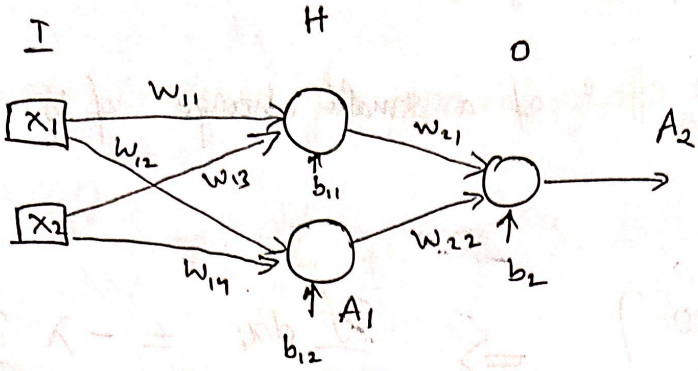
XOR problem using Multi layered Neural Network.



We consider $X = [x_1 \; x_2]^T$ as the input bits for a two layered neural network. Weights of the first layer is

~~$W1 = [w_{11} \; w_{12} \; w_{13} \; w$~~

$W1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{13} & w_{14} \end{bmatrix}$ and biases, $b_1 = \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix}$. Similarly, the weights and biases of second or output layer is $W_2 = \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}$ & b_2.

We use sigmoid as the activation function for introducing non linearity in the network and optimise the network parameter by minimizing "binary crossentropy loss" defined by

$$J(Y, \hat{Y}) = -\left[Y \log_2(\hat{Y}) + (1-Y)\log_2(1-\hat{Y})\right]$$

where : $Y \rightarrow$ desired output

$\hat{Y} \rightarrow$ actual output

Forward Propagation :

First layer:

$$Y_1 = \langle X, W1 \rangle + b_1$$

$$A_1 = \sigma(Y_1)$$

output layer:

$$Y_2 = \langle A_1, W_2 \rangle + b_2$$

$$A_2 = \sigma(Y_2) \text{ [computed output]}$$

$\langle \cdot, \cdot \rangle \rightarrow$ dot product

$\sigma \rightarrow$ sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

If we assume Y as the desired output, then the loss is

$$J(Y, A_2) = -\left[Y \log(A_2 + \epsilon) + (1-Y)\log(1 - A_2 + \epsilon)\right]$$

(we add $\epsilon \leq 10^{-5}$ since if $A_2 = 0$, then $\log(0) \to \infty$ and the program crashes. Therefore, adding ϵ provides numerical stability).

Back propagation:

learning rate, $lr = 0.05$

we update the weights and biases of the network using gradient descent, i.e.,

$$w_i = w_i - lr \cdot \frac{\partial J}{\partial w_i} \quad \text{and} \quad b_i = b_i - lr \cdot \frac{\partial J}{\partial b_i}$$

$$i = 1, 2$$

Computing the gradients for the output layer.

$$\therefore \frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial A_2} \cdot \frac{\partial A_2}{\partial Y_2} \cdot \frac{\partial Y_2}{\partial W_2}$$

$$\frac{\partial J}{\partial W_2} = \frac{(A_2 - Y)}{A_2(1 - A_2)} \cdot A_2(1 - A_2) \cdot A_1 = (A_2 - Y) \cdot A_1$$

$$\therefore \boxed{\frac{\partial J}{\partial W_2} = (A_2 - Y) \cdot A_1}$$

$$\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial A_2} \cdot \frac{\partial A_2}{\partial Y_2} \cdot \frac{\partial Y_2}{\partial b_2} = \frac{(A_2 - Y)}{A_2(1 - A_2)} \cdot A_2(1 - A_2) \quad (1)$$

$$\boxed{\frac{\partial J}{\partial b_2} = (A_2 - Y)}$$

Computing gradients for the hidden layer.

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial A_2} \cdot \frac{\partial A_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Y_1} \cdot \frac{\partial Y_1}{\partial W_1}$$

$$\frac{\partial J}{\partial W_1} = \frac{(A_2 - Y)}{A_2(1-A_2)} \cdot A_2(1-A_2) \cdot W_2 \cdot A_1(1-A_1) \cdot X$$

$$\therefore \boxed{\frac{\partial J}{\partial W_1} = \langle (A_2 - Y), W_2 \rangle \cdot A_1(1-A_1) \cdot X}$$

Similiarly,

$$\boxed{\frac{\partial J}{\partial b_1} = \langle (A_2 - Y), W_2 \rangle \cdot A_1(1-A_1)}$$

We then iteratively update weights and biases until the loss converges to zero.