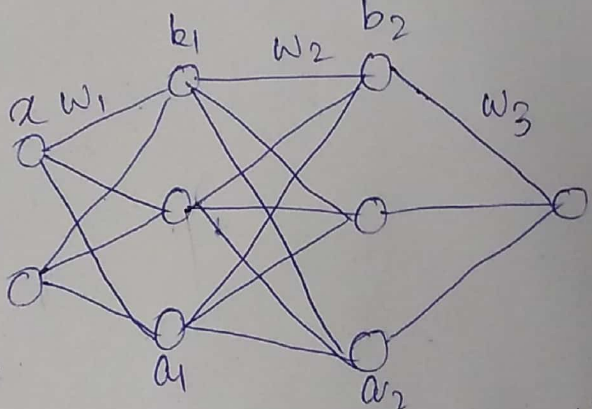
1)

No, a neural net of arbitary depth using just linear functions can't be used to model the XOR truth table, because a neural net of any depth with the linear activation functions is same as linear classifier. As XOR truth table isn't linearly separable so neural net with linear activation fn c. can't model the XOR truth table.



Let a_1 and a_2 be linear activation fnc.

Then i/p of layer-2 $= x w_1$

o/p of layer-2 $= a_1(w \cdot x_1) = c \cdot x w_1 + b$

i/p for layer 3 $= c \cdot x \cdot w_1 \cdot w_2 + w_2 b$

o/p for layer 3 $= a_2(c \cdot x \cdot w_1 \cdot w_2 + w_2 b)$

$= c_1 c x w_1 w_2 + c_1 w_2 b + b_2$

o/p for layer 4 $= c_1 c\, x\, w_1 x_2 + c_1 w_2 b + b_2$

$= x \cdot w^* + b^*$

$w^* = W_1 W_2 C_1 C_0$

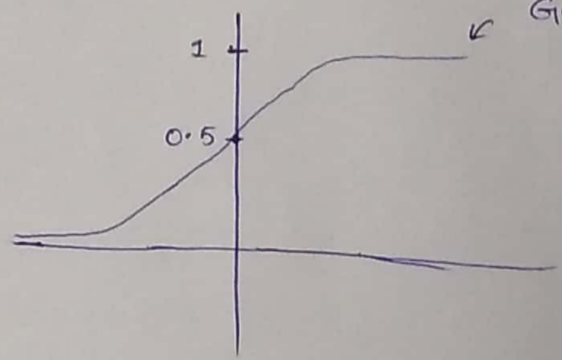
$b^* = C_1 W_1 b_1 b_2$

So, instead of using 2 hidden layers with linear activation func, single hidden layer with $W = w^*$ would have done same result.

$\therefore$ It's similar to linear regression discussed in class $f(x, w_1) = x w^* + b^*$ which basically represents ~~so~~ linear classifer.

Scanned by CamScanner

2) The reason behind the model wasn't able to train because when $\sigma(z)$ gets flattened when value is near 1, so value of $\sigma'(z)$ become very less (zero gradient)



Due to zero gradient there'll be no update in w.

If x uses ReLU instead of sigmoid the zero gradient problem will be vanished, but the problem will be data explosion. which at a given time weights can be increased exponentially.

No updation for negative weights also another problem for ReLU ($\min \nabla_z f(z)$ can be 0)

For sigmoid we should use scaling for the i/p.

For ReLU we should normalize the data.

3)

The squared error cost function is

$$c = \frac{1}{2}(y-a)^2$$

$$\therefore \frac{\partial c}{\partial w} = (a-y)\cdot\sigma'(z)\cdot z$$

$$\frac{\partial c}{\partial b} = (a-y)\,\sigma'(z)$$

where w = weight
b = bias

Now the problem arises due to the term $\sigma'(z)$ in the partial derivatives. If we follow the graph of sigmoid function, we can see that for any node graph gets very flatened when the value (o/p) is near about 1, due to which change in $\sigma(z)$ (or $\sigma'(z)$) will be really really small, so final o/p becomes small.

Now for the cross entropy cost function is

$$c = -[y \log y + (1-y)\log(1-y)]$$

By taking partial derivative of c w.r.t w

$$\frac{\partial c}{\partial w} = -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\cdot\sigma'(z)\cdot x$$

$$= -\left[\frac{y - \sigma(z)\cdot y - \sigma(z) + \sigma(z)\cdot y}{\sigma(z)(1-\sigma(z))}\right]\cdot\sigma'(z)\cdot x$$

$$\therefore \frac{\partial c}{\partial w} = -\left[\frac{y-\sigma(z)}{\sigma(z)(1-\sigma(z))}\right] \cdot \sigma'(z) \cdot x$$

$$= \frac{\sigma(z)-y}{\sigma(z)(1-\sigma(z))} \cdot \sigma'(z) \cdot x$$

As we know $\sigma'(z) = \sigma(z)(1-\sigma(z))$

$$\therefore \frac{\partial c}{\partial w} = x(\sigma(z)-y)$$

Similarly $\frac{\partial c}{\partial b} = \sigma(z) - y$

Above mentioned partial derivatives cancel out the term $\sigma'(z)$ or the change in the o/p, so even near if the Graph gets flattened output won't minimized due to $\sigma'(z)$ term.