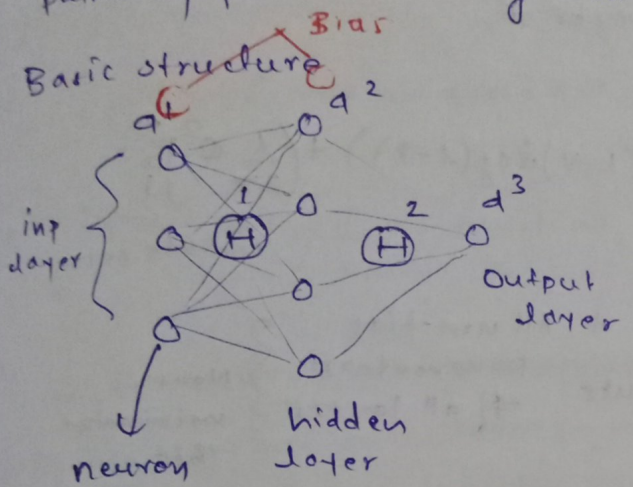
* Neural Networks *

For non linear hypothesis-

How many more secondary or primary features should we add to get perfect equation for logistic regression.

We will see that neural networks generate an easy pathway for implementing non-linear - complex hypothesis

Basic structure



Neurons after 1st layer take only binary values.

Step 1:

a. Calculate a_1^2 using a_k^1

$\therefore z^2 = (\Theta^1 * (a^1 + \text{bias } a_0^1))$

$a_b^2 = g(z^2)$ ⇓ z^2

b. Calculate a^3 using a_k^2

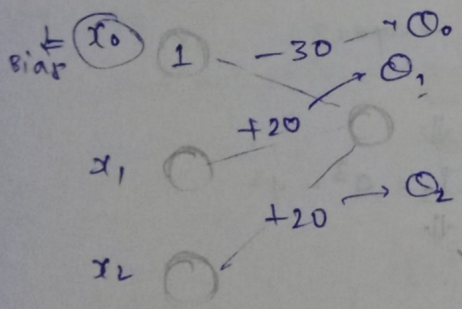
$z^3 = \Theta^2 * (a^2 + \text{bias } a_0^2)$

$h_\Theta(x^1)$ $\{a^3 = g(z^3)$

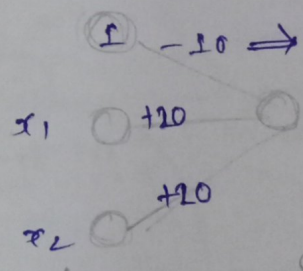
Hypothesis.

1) Why is bias so necessary??

AND Gate implementaion using NN

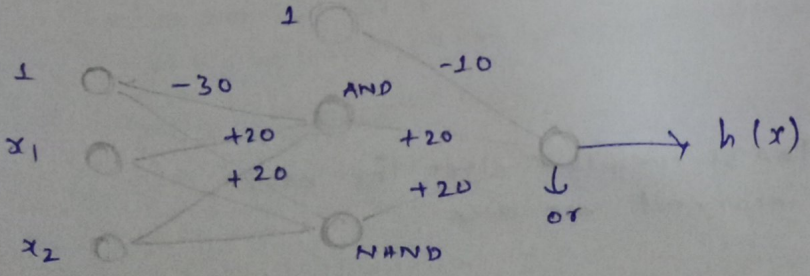


Or gate implementation using NN



⟹ Just adding a bias makes two gates possible

XNOR → nand } some gate



} Intuition behing NN

* Multiclass NN

Test sets are like $\left(x^1 = \begin{bmatrix} 1 \\ \vdots \\ n \end{bmatrix}, y \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) \ldots$

↓ multiclass

* Cost function for Neural Networks *

* $J(\Theta)$ for logistic

$$J(\Theta) = \frac{-1}{m} \sum_{1}^{m} \left(y \log(P) + (1-y)\log(1-P) \right) + \left(\frac{\lambda}{2m} \Theta^2 \right)_{1}^{n}$$

↓ exclude zero

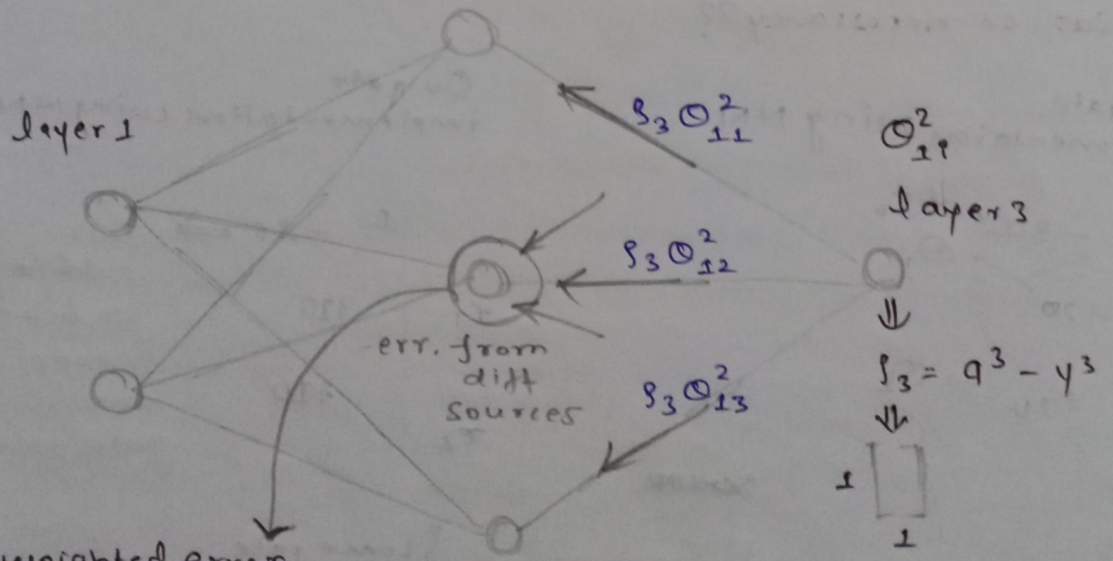
$$J(\Theta) = \left(\frac{-1}{m} * \text{sum} \right) * \text{for all K outputs} + \text{All non-bias parameters of all layers}$$

} Now minimize this.

---

* BackPropogation Algorithm *

layer 2 $= \Theta^1_{1i}, \Theta^1_{2i}, \Theta^1_{3i}$



The weighted error is summed and multiplied with gradient of a^2_2 and make up a new error for this new layer.

$$\delta_2 = \{\Theta^2\}^T * \delta_3 * a^{(2)} . * (1 - a^{(2)})$$

The steeper the slope the more incorrect we are

ज्ञान, "

↑ sigmoid

In logistic regression we used $\frac{1}{2m}$ and in linear regression we used $(\text{error})^2$ } To get convex curve.

∴ when we use Sigmoid function as an activation function we use $\frac{1}{m}$ for cost } To get nearly convex curve.

$J(\theta)$ → we have to find local & global minimum

θ^3 / θ^4 → input parameters / weights

$\theta_{10}^{(2)}$

$\theta^{(2)}_{11} \approx \theta^2$

∴ We use gradient descent algorithm

⇒ fminunc { @ costfunction ↑ which recurrently needs gradient and cost at each } the 'maxiter' part.

∴ For neural network what can be the gradient descent.

↳ Remember that calculating gradient descent was an easy task for logistic regression where this was the framework

bias { θ_0

θ_1

θ_2

θ_3

inp { } out } i.e by directly finding derivative } cost function.

". ज्ञान समाप्त.

और ज्ञान, "

But what in case } derivative for $\theta^{(1)}$

$\theta^{(1)}$ a_1^2

And we have to find gradient to continously feed it

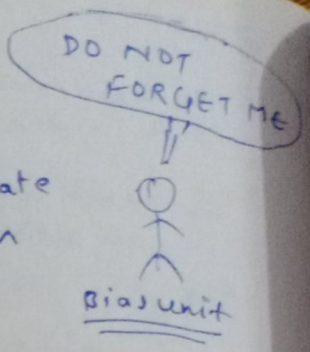
⇓

fminunc.

Note: We could have use $\frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$ also for all θ but it would have been computationally expensive.

". ज्ञान समाप्त

Step by step tutorial to coalculate
The gradient.



for all training sets 1 by one

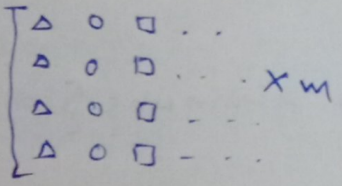
{ a. Perform forward pass to calculate the value of activation function at each layer

'''
Assuming layer{row} by row input set
Activation function as column vector
Theta is also a row by row input set
'''

a-1 = X

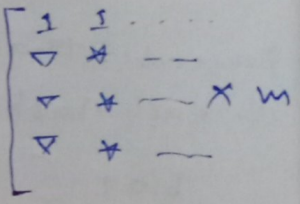
a-2 = g(Theta * X')

a-2 = ~~a-~~
[ones(1,m);
a-2]



a-3 = g(Theat_2 * a-2)

a-3 = [ones(1,m);
a-3]



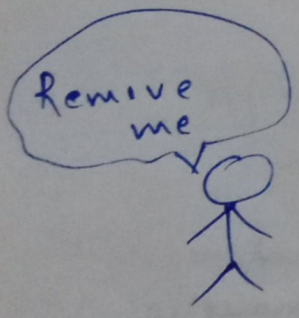
Hence $h_\Theta(x)$ is obtained.

b. ''' Assuming 4 layer'''
calculate δ error using the formulaes

$\delta^4 = a\text{-}4 - Y$

$\delta^3 = (\Theta^3)^T \delta^4 .* a\text{-}3 .* (1 - a\text{-}3)$

$\delta^2 = (\Theta^2)^T \delta^3 .* a\text{-}2 .* (1 - a\text{-}2)$



} $\delta^3 = \delta^3[2:end]$

c. Just a minute step behind gradient.

$\Delta^{\delta} = \Delta^{\delta} + \delta^{\delta} * (a^{\delta})^T$

$\Delta^1 = \Delta^1 + \delta^2$