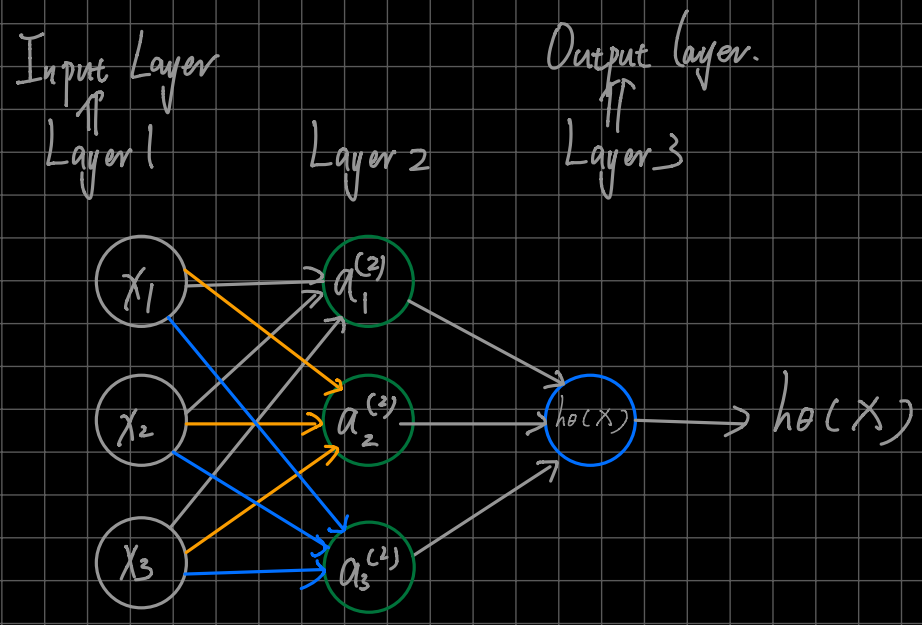
Neural Network:



$$a_1^{(2)} = g(\theta_{10}^{(1)} \cdot x_0 + \theta_{11}^{(1)} \cdot x_1 + \theta_{12}^{(1)} \cdot x_2 + \theta_{13}^{(1)} \cdot x_3)$$

$$a^{(2)} = g(\theta_{20}^{(1)} \cdot x_0 + \theta_{21}^{(1)} \cdot x_1 + \theta_{22}^{(1)} \cdot x_2 + \theta_{23}^{(1)} \cdot x_3)$$

$$a^{(2)} = g(\theta_{30}^{(1)} \cdot x_0 + \theta_{31}^{(1)} \cdot x_1 + \theta_{32}^{(1)} \cdot x_2 + \theta_{33}^{(1)} \cdot x_3)$$

bias.

(added neuron) $h_\theta(x) = g(\theta_{10}^{(2)} \cdot a_0 + \theta_{11}^{(2)} \cdot a_1 + \theta_{12}^{(2)} \cdot a_2 + \theta_{13}^{(2)} a_3)$

k+1th layers

△ $a_i^{(k+1)} = g(\theta_{i0}^{(k)} \cdot a_0^{(k)} + \theta_{i1}^{(k)} \cdot a_1^{(k)} \cdots + \theta_{im}^{(k)} \cdot a_m^{(k)})$

i: ith neuron

⋮

$$a_n = g\left(\theta_{n0}^{(k)} \cdot a_0^{(k)} + \theta_{n1}^{(k)} \cdot a_1^{(k)} + \cdots + \theta_{nm}^{(k)} \cdot a_m^{(k)}\right)$$

Thus, the $(k+1)$th layer has a parameter matrix θ of size $(m+1) \cdot n$

m: kth layer neuron number

n: $(k+1)$th layer neuron number

Vector Implementation:

$$x = \begin{bmatrix} x_0 \\ \vdots \\ x_m \end{bmatrix}, \quad z^{(k)} = \begin{bmatrix} z_1^{(k)} \\ z_2^{(k)} \\ \vdots \\ z_n^{(k)} \end{bmatrix}$$

$a_1^{(k)} = g(z_1^{(k)})$

$a_2^{(k)} = g(z_2^{(k)})$

① $z^{(k)} = \theta^{(k-1)} \cdot z^{(k-1)}$

$z^{(k)}$: $n \times 1$; $\theta^{(k-1)}$: $n \times (m+1)$; $z^{(k-1)}$: e.g. x, $(1+m) \times 1$

② $a^{(k)} = g(z^{(k)})$

③ $z^{(k+1)} = \theta^{(k)} \cdot a^{(k)}$

$\Rightarrow$ ① ② ③

$a_0^{(k)} = 1$

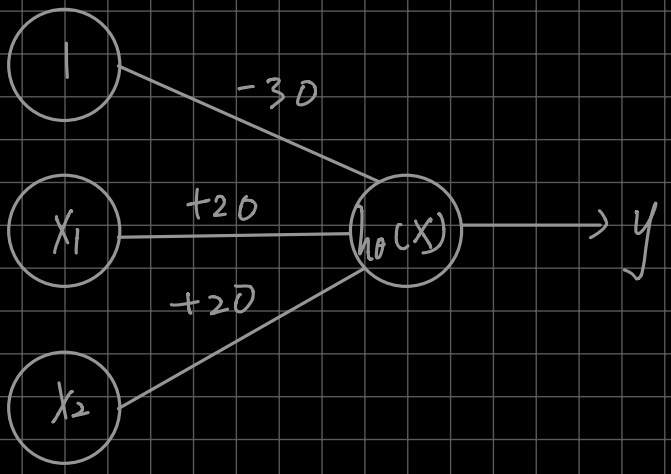
$a^{(k+1)} = g^{(k+1)}(z^{(k+1)})$ ①

$\Rightarrow$ ②

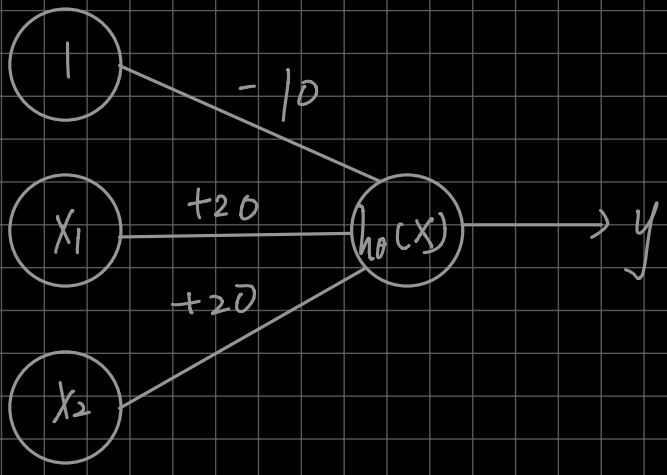
$a_0^{(k+1)} = 1$ ③

Examples and intuition:

x_1 AND x_2



x_1 OR x_2

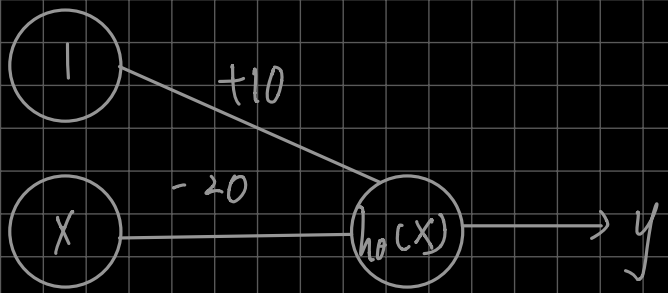


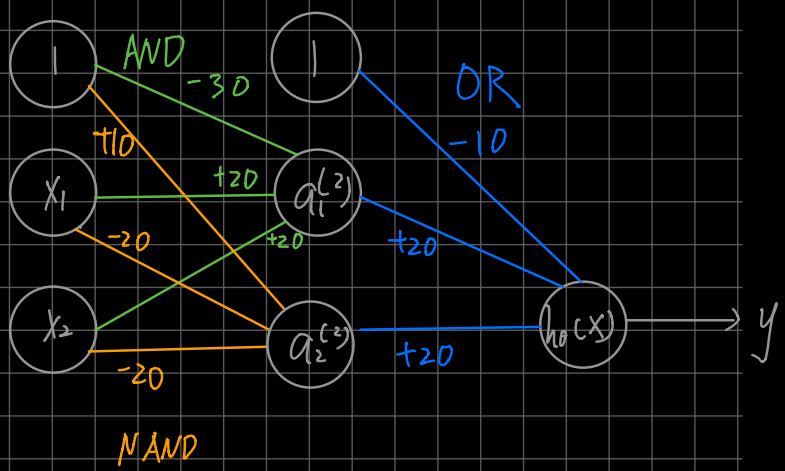
x_1 NOR x_2 $\Rightarrow$

Using Decomposition and combination:

x_1 NOR $x_2 \Leftrightarrow$ (x_1 AND x_2) OR ((NOT x_1) AND (NOT x_2))

NOT x





NAND

△ We can know that every logic expression can be implemented using Neural Network. We just need to take them apart to form small neural networks and put them together to form a big neural network.

3 things you need to know about neural network:

- In API, you need to unrolling parameters.
  Input || return values
- Gradient checking: $\frac{\partial J(\theta)}{\partial \theta_j} \approx \frac{J(\theta_i + \varepsilon, \theta') - J(\theta_i - \varepsilon, \theta')}{2\varepsilon}$ → $\theta_i + \theta' = \theta$
- Random initialization of parameters ⇒ avoid symmetric update