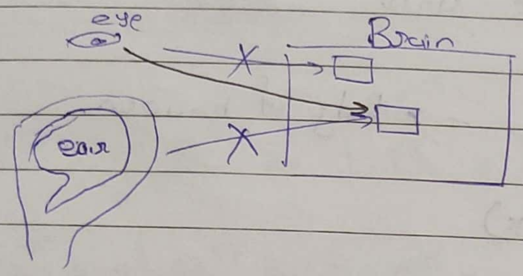
☆ Week 4 ☆

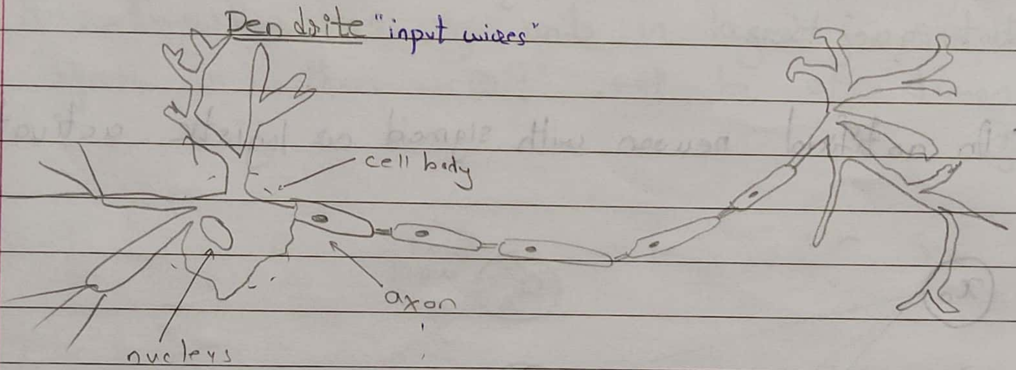
neurons & brain (nural nets)

--@ Model Representation I — — —



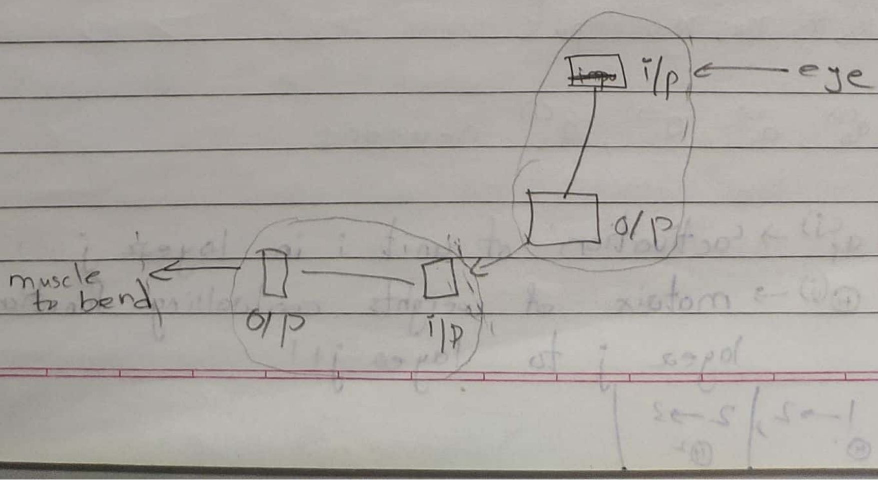



- If we ~~dissc~~ disconnet ear & auditory cortex.
- Connect eye to auditory cortex. (ac)
  → ac will learn to see.
  → This tell there should be only one algorithm to train brain.




"output wire"
send message to other neuron

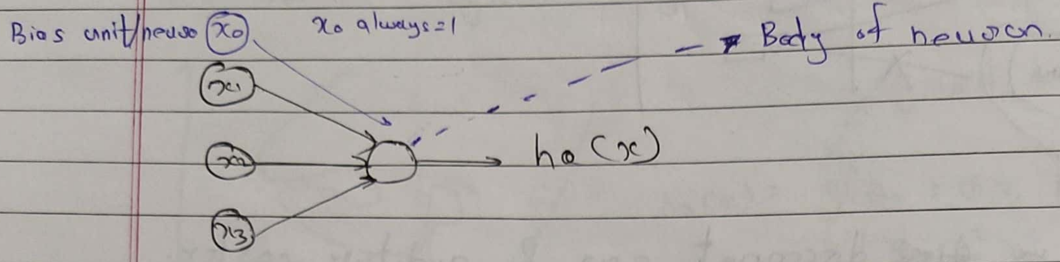
- Takes input.
- computation
- output via axon to other neurons in brain. ---> other neuron input



$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ = parameters = weights



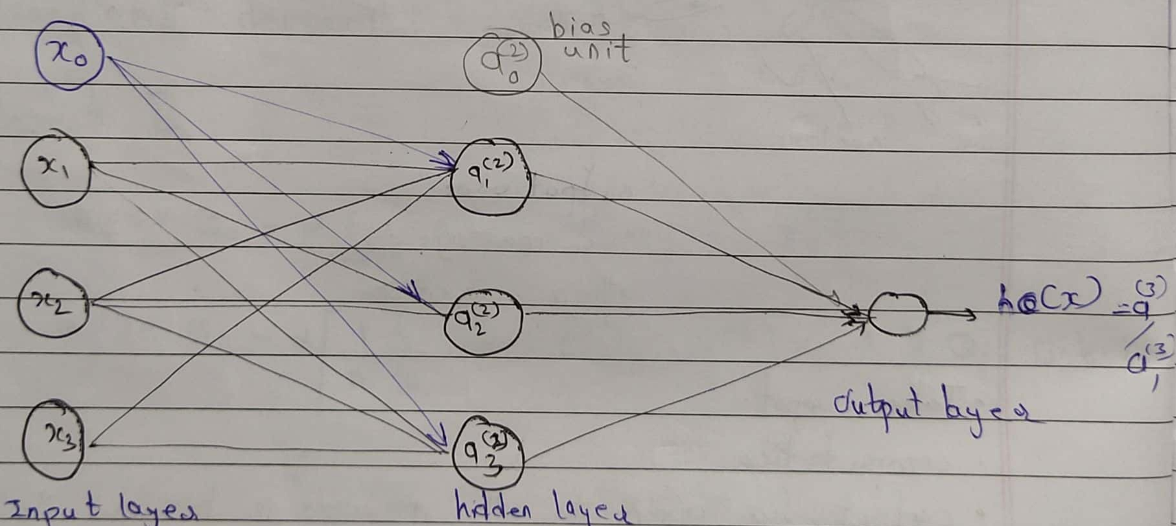
- Neuron model: logistic unit




$h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}}$

$x_0$ is drawn according to notation simplicity of that question.

→ An artificial neuron with sigmoid or logistic activation fn.




$x_0, x_1, x_2, x_3$ inputs

$a_0^{(2)}, a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$ neurons

$a_i^{(j)}$ → 'activation' of unit $i$ in layer $j$

$\Theta^{(j)}$ → matrix of weights controlling function mapping from layer $j$ to layer $j+1$

| 1 → 2, $\Theta^1$ | 2 → 3 $\Theta^2$ |

→ layer (j)

$\Theta_{10}^{(1)} x_0$

layer (j+1) going

(j) layer comming ⟷ $j^{th}$ layer $0^{th}$ is bias hidden term

## ☆ Neural Network

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

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has $s_j$ units in layer $j$, $s_{j+1}$ units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension

$$s_{j+1} \times (s_j + 1)$$

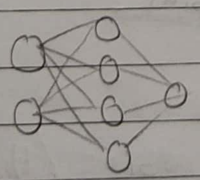
2nd — 3 — $\Theta^{(1)}$ — 1st, 3. — $3 \times 4$

$\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$

3rd — 3 — $\Theta^{(2)}$ — 2nd, 1 — $1 \times 4$

$= a_1^{(3)}$

**Que**

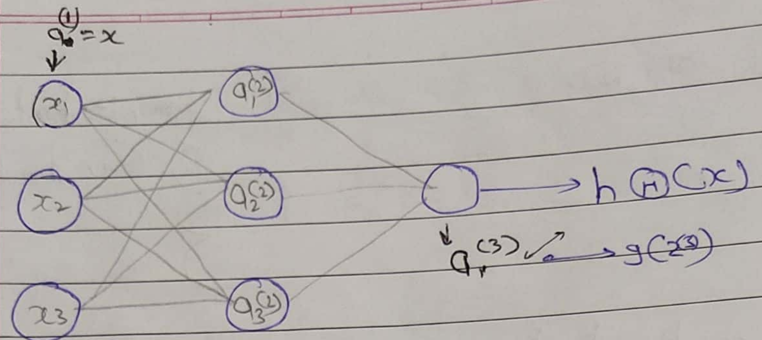


$s_1 = 2$ $\quad s_2 = 4$ $\qquad \Theta^{(1)} \rightarrow \mathbb{R}^{4 \times 3}$

— @ Model Representation II —



Coursera continued




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

$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3$

$a_1^{(2)} = g(z_1^{(2)})$

- $a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \to z_2^{(2)}$
- $a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \to z_3^{(2)}$
- $h_\Theta(x) = g(\Theta_{30}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{33}^{(1)} x_3) \to z^{(3)}$

- vectorization

$x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$z^{(2)} = \Theta^{(1)} x \longrightarrow z^{(2)} = \Theta^{(1)} a^{(1)}$ ← activations in input layer

$a^{(2)} = g(z^{(2)})$ $\quad (z^{(2)} = \Theta^{(1)} a^{(1)})$ vectorization

$a^{(2)} \to \mathbb{R}^3$, $g(z^{(2)}) \to \mathbb{R}^3$

This sigmoid fn is applied element wise to each $z^{(2)}$'s elements

Add one more term $a_0^{(2)} = 1 \xrightarrow{\text{now}} a^{(2)} \in \mathbb{R}^4$

$z^{(3)} = \Theta^{(2)} a^{(2)}$

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

Architecture :- way neural network are connected



## Forward propagation.

- Here we start by activation of input units. (randomly)

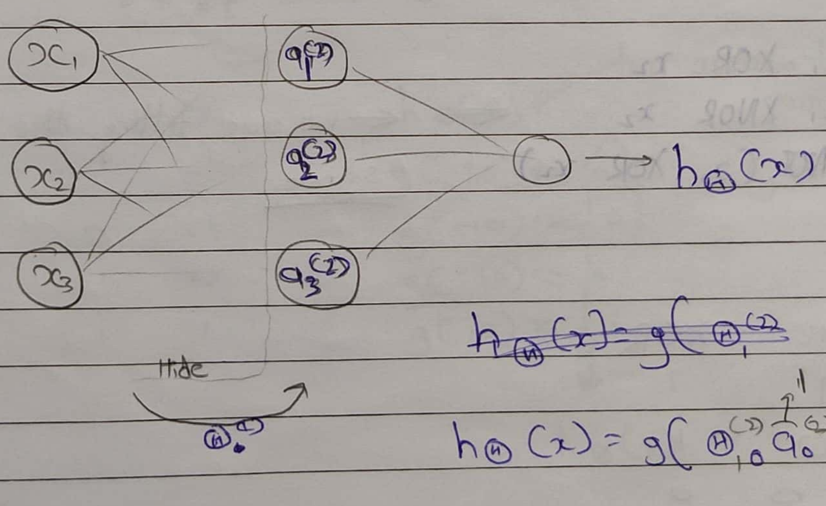
$\downarrow$ forward propagate to

hidden layer (Here we compute activations of hidden layer)

$\downarrow$ forward propagate to

output layer (Here we compute activation of output layer)

$z^{(3)}$




~~$h_\Theta(x) = g(\Theta_1^{(2)}$~~

layer

$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(1)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

Just like logistic reg$^{n}$:

$\theta \rightarrow \Theta$

$x_1, x_2, x_3 \rightarrow a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$

neural network is used to learn complex non-linear hypothesis.

Page No: 49

Date: 22/6/19

1st day in pune

## Examples and Intutions I

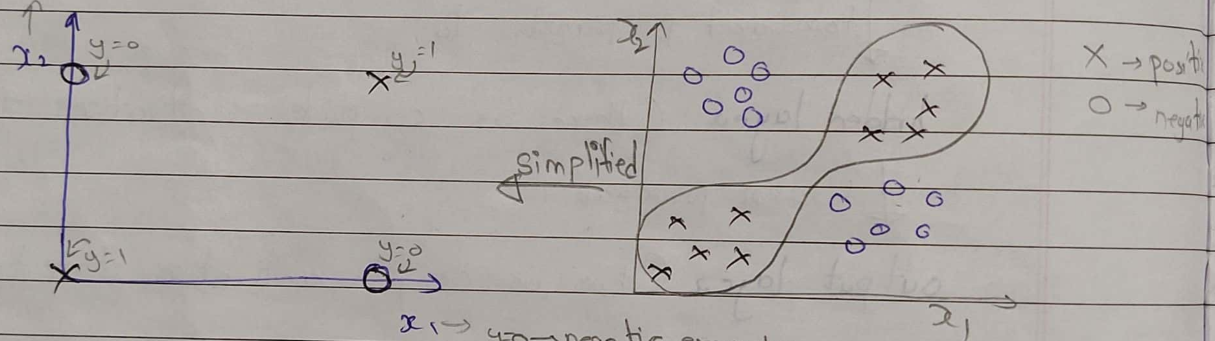
Raw features → $x_1, x_2, x_3$

NN learn on features $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$

## Applications

e.g. 1) Non-linear classification example XOR / XNOR

$x_1, x_2$ are binary $\in \{0, 1\}$




y=0 → negative examples
y=1 → positive examples

$y = x_1$ XOR $x_2$

$x_1$ XNOR $x_2$ ⇐ ← we are fitting this example

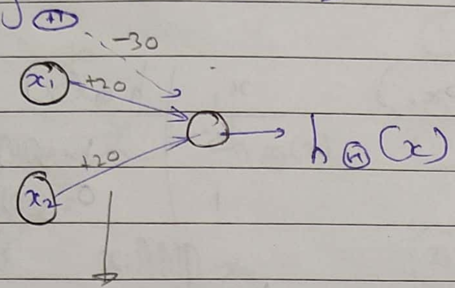
NOT ($x_1$ XOR $x_2$) same

| $x_1$ | $x_2$ | XOR |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

eg.) 1.1 > Simple eg. AND (a)

$x_1, x_2 \in \{0,1\}$

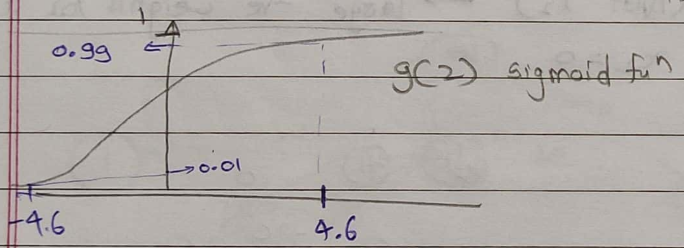
$y = x_1$ AND $x_2$




assigning the parameters

$$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$$

$\Theta^{(1)}_{10}$ $\quad \Theta^{(1)}_{11}$ $\quad \Theta^{(1)}_{12}$



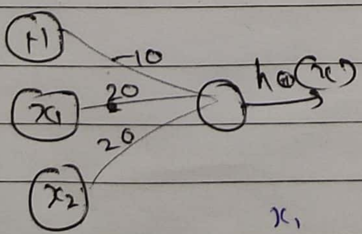

for assigned parameter

| $x_1$ | $x_2$ | $h_\Theta(x)$ |
|---|---|---|
| 0 | 0 | $g(-30) \approx 0$ |
| 0 | 1 | $g(-10) \approx 0$ |
| 1 | 0 | $g(-10) \approx 0$ |
| 1 | 1 | $g(10) \approx 1$ |

— (by sigmoid fn)

$h_\Theta(x) \approx x_1$ AND $x_2$.

que




which is this logical fn (Hint: Draw truth table)

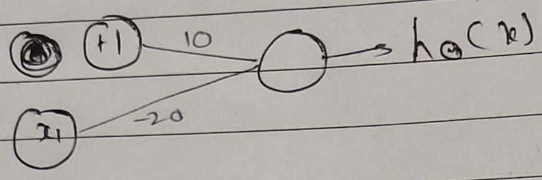
| $x_1$ | $x_2$ | $h_\Theta(x)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

fn

$x_1$ OR $x_2$

## -- ⓐ Examples & Intutions II

e.g) Negation → ⓑ




$h_\Theta(x) = g(10 - 20x_1)$

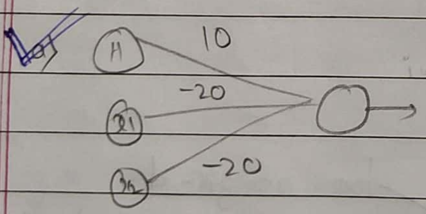
| $x_1$ | $h_\Theta(x)$ |
|---|---|
| 0 | $1 \approx g(10)$ |
| 1 | $0 \approx g(-10)$ |

The Idea is put large negative weight in front of the variable you want to negate.

→ Guessing? weights !!..

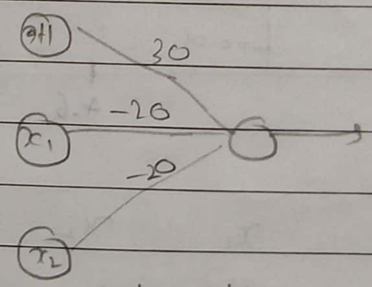
e.g.) (NOT $x_1$) AND (NOT $x_2$) ⓒ → large -ve weights for $x_1, x_2$ to negate
→ = 1 iff $x_1 = x_2 = 0$

a)



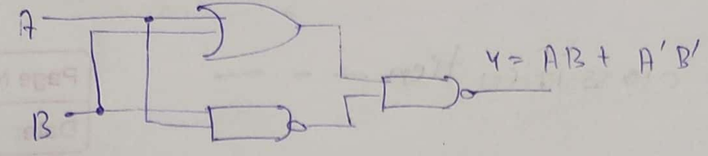

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | $g(2) = 0$ |
| 0 | 1 | $g(2) = 0$ |
| 0 | 0 | 1 |

b)



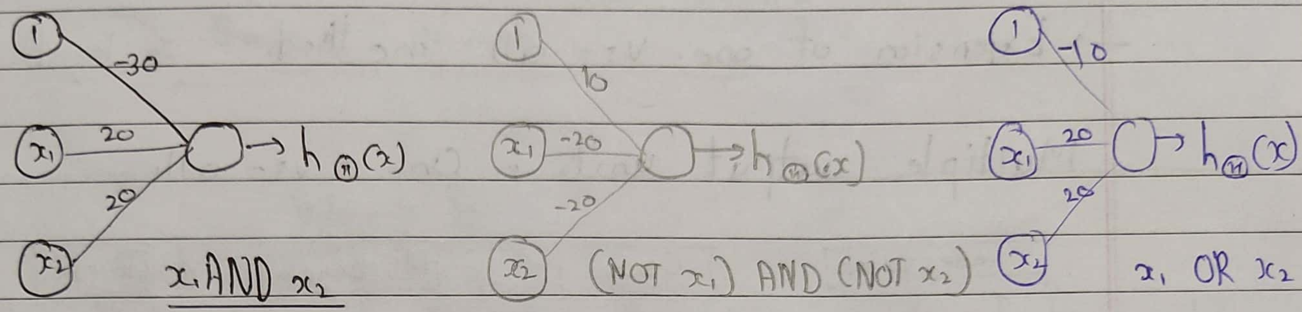

| | | |
|---|---|---|
| 1 | 1 | 0 |
| ~~1~~ | 0 | $g(2) = 1$ |
| ~~0~~ | 1 | $g(2) = 1$ |
| 0 | 0 | 1 |

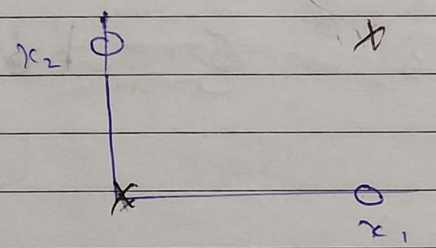
| A | B | A XOR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



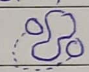

putting ⓐ,ⓑ,ⓒ we should be able to compute $x_1$ XNOR $x_2$

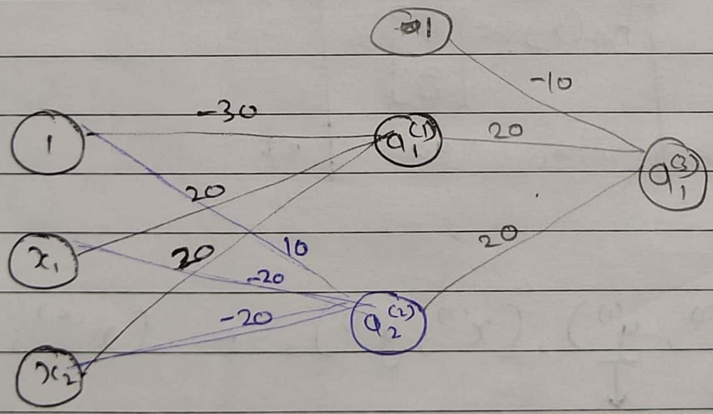
Putting together




Here, we will need non-linear decision boundary →






| $x_1$ | $x_2$ | $a_1^{(2)}$ | $a_2^{(2)}$ | $h_\Theta(x)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

$h_\Theta(x) = 1$ exactly at these two locations.

* thus we, have done with finding non-linear ~~hypothesis~~ decicion boundry.

--@ Multiclass classification

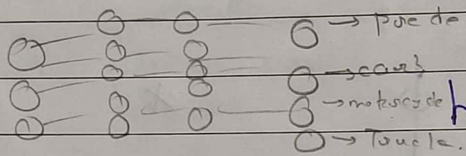
# ✠ Multiclass classification.

e.g.) Reading / classifying Handwritten digit.

→ Extension of one-vs-all method → In logistic regressionⁿ.

Multiple output units : One-vs-all.

| Paedestrain | | Car | | Motorcycle | | Truck |




$h_\theta(x) \in \mathbb{R}^4$.

We want,

| $h_\theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ | $h_\theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ | |
|---|---|---|
| when pedestrain | when car | etc. |

Now → Training set : $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})$

$x^{(1)}$ ↓ Image, $y^{(1)}$ ↓ vector → $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ / $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$

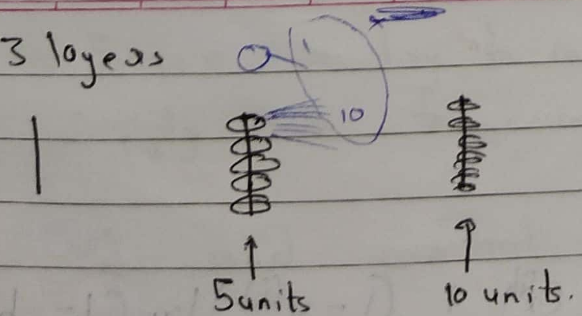
pedestrain / car

previously → ~~$y \in \{1, 2, 3, 4\}$~~

$$h_\theta(x^{(i)}) \approx y^{(i)}$$

$\mathbb{R}^4$

**Que** 3 layers




How many Elements in $H^{(2)}$ = 60 = 10×6