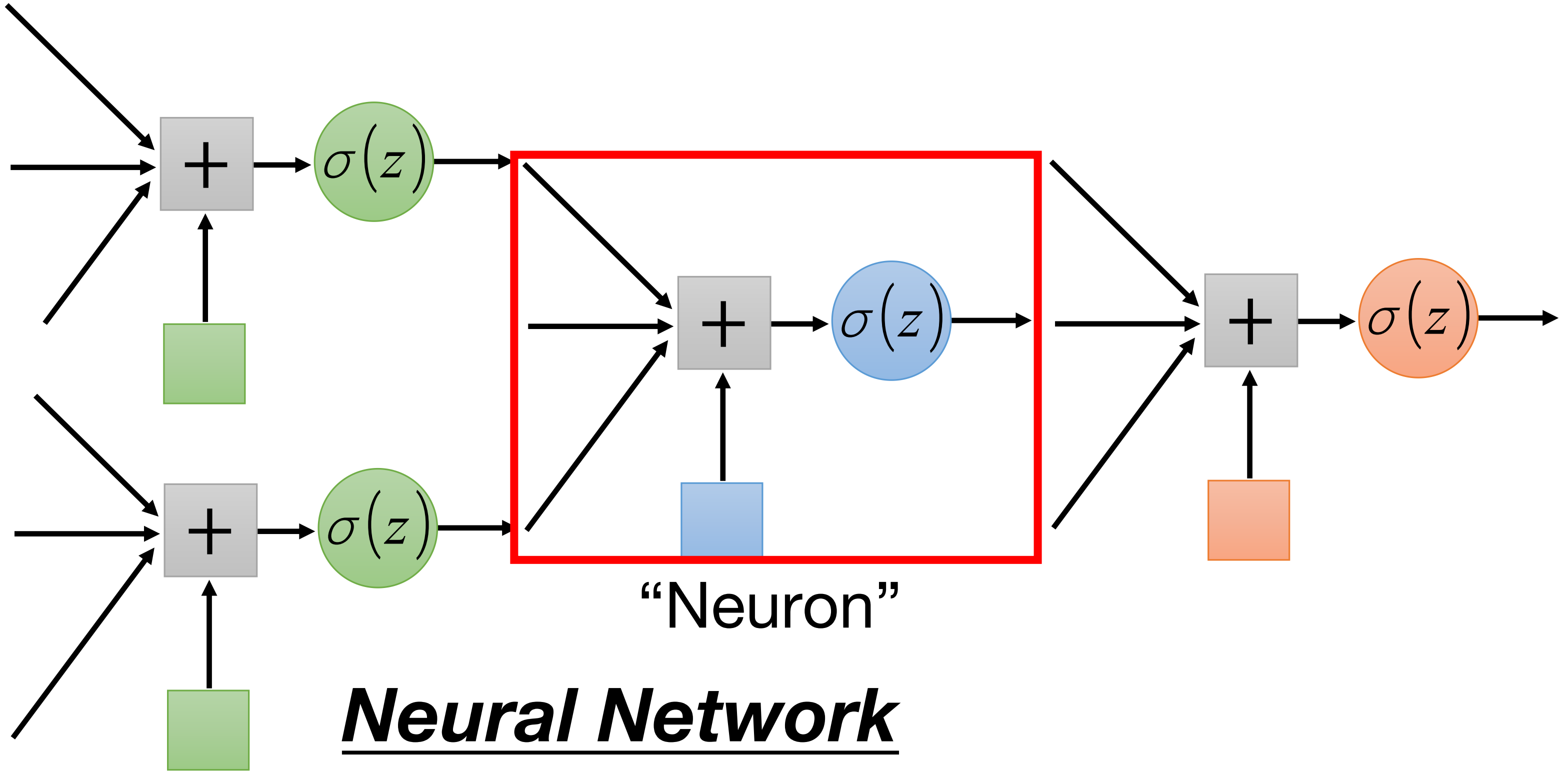
Neural Networks (NN)

Content

- ■Neural Network representation
  - Model representation
  - NN for logic expressions

- ■Neural Network learning
  - Cost function
  - Backpropagation (BP) algorithm

Neural Network

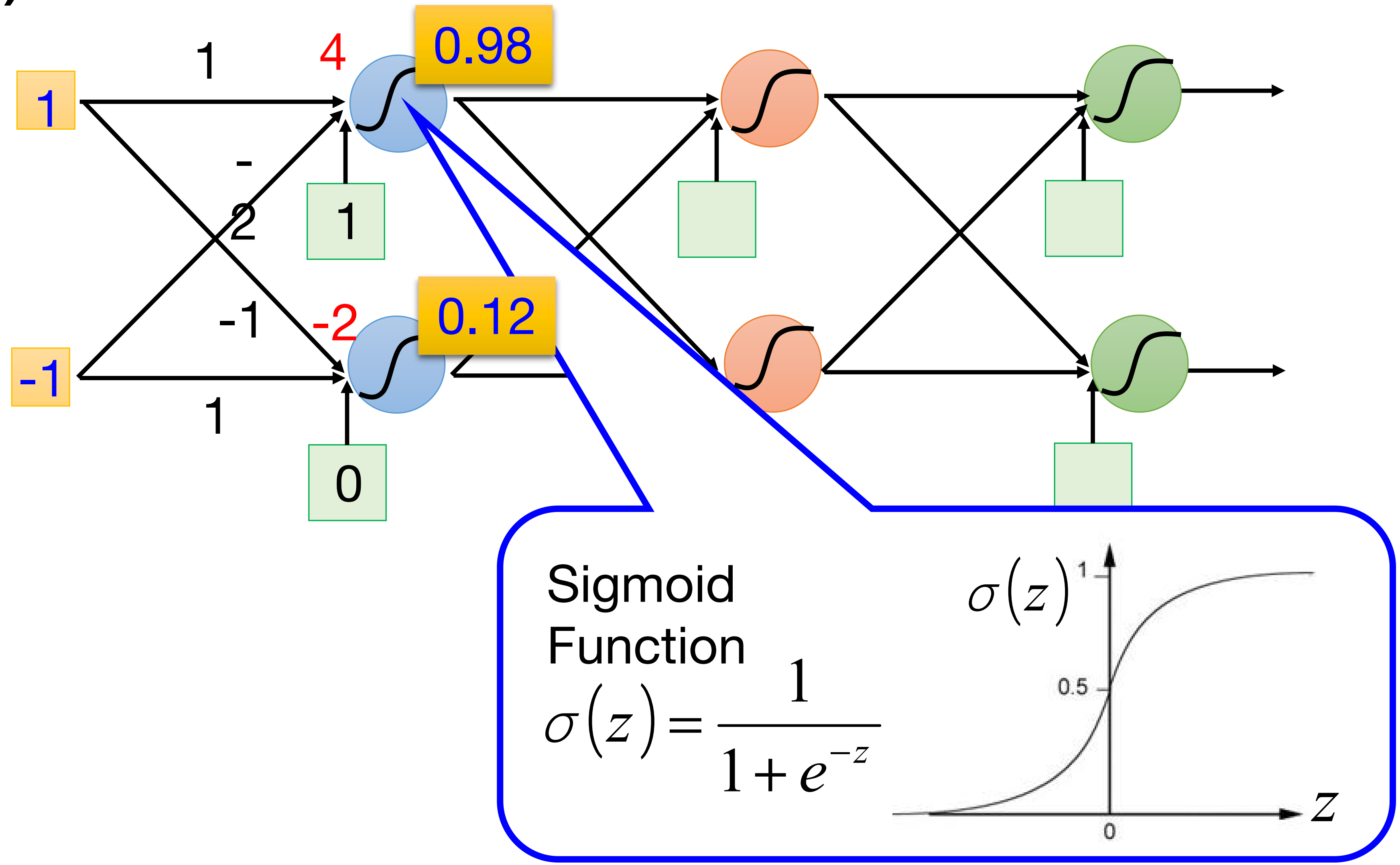


<u>Neural Network</u>

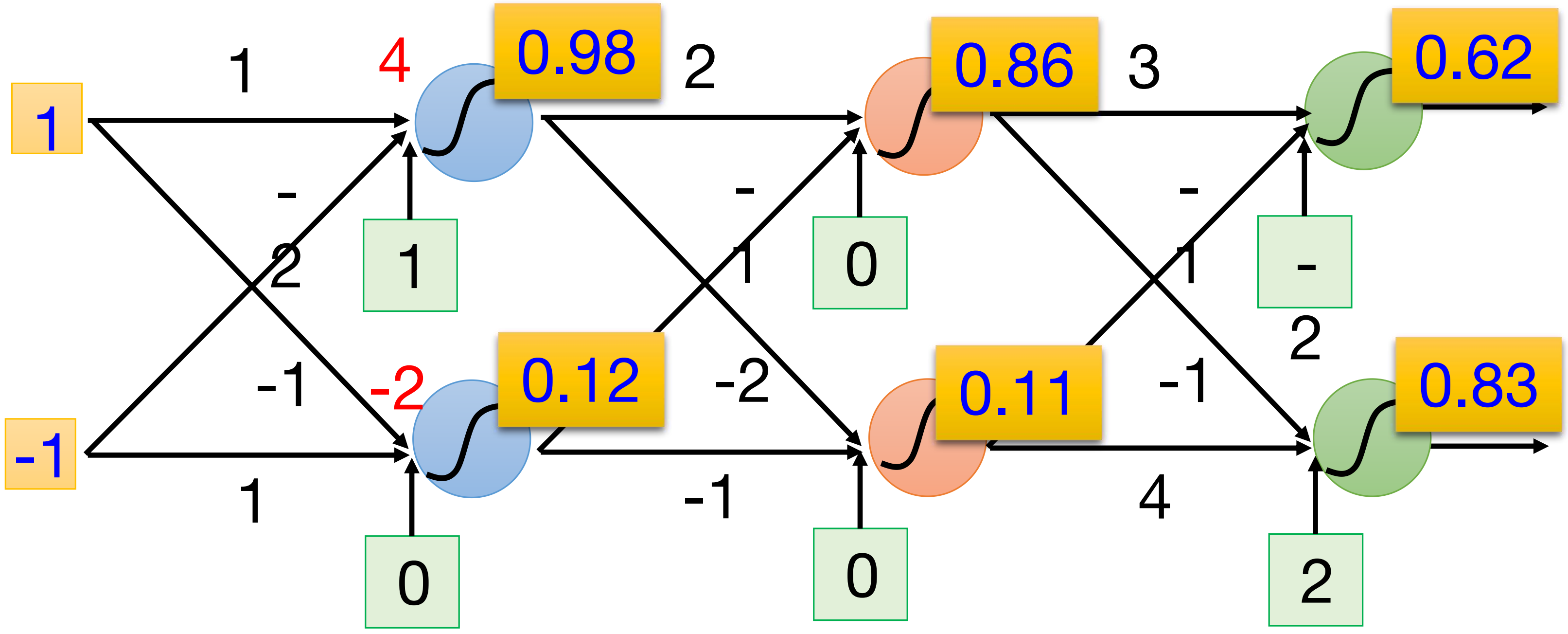
Different connection leads to different network structures

Network parameter θ: all the weights and biases in the “neurons”

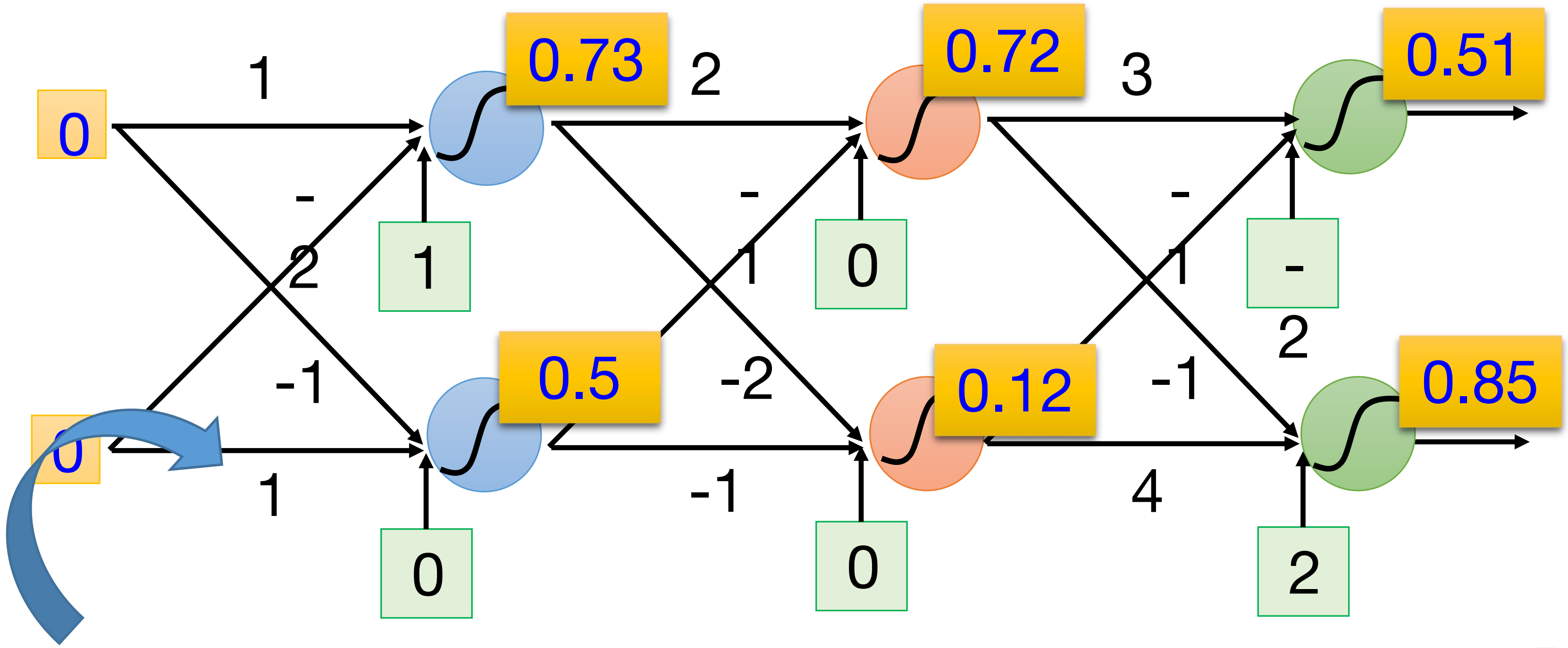
Fully Connected Feedforward Network (FCN)



Fully Connected Feedforward Network



Fully Connected Feedforward Network

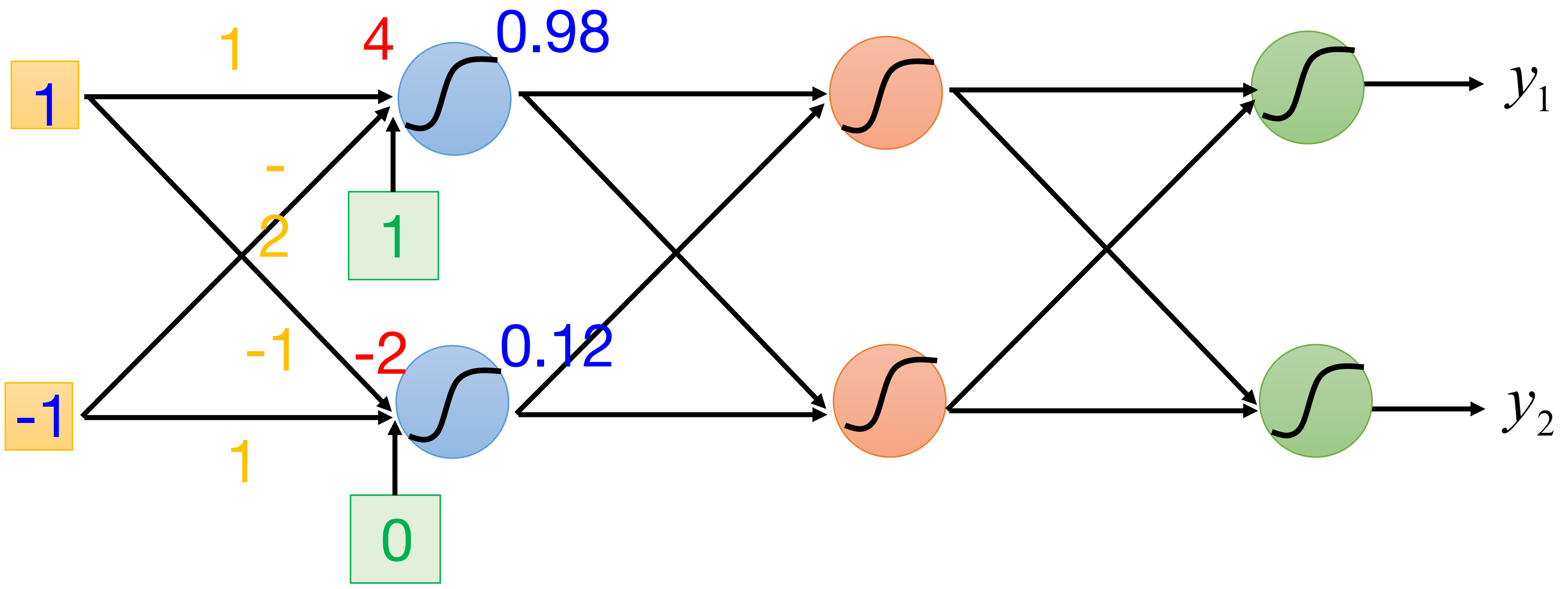


This is a function.
Input vector, output vector

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix} \qquad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

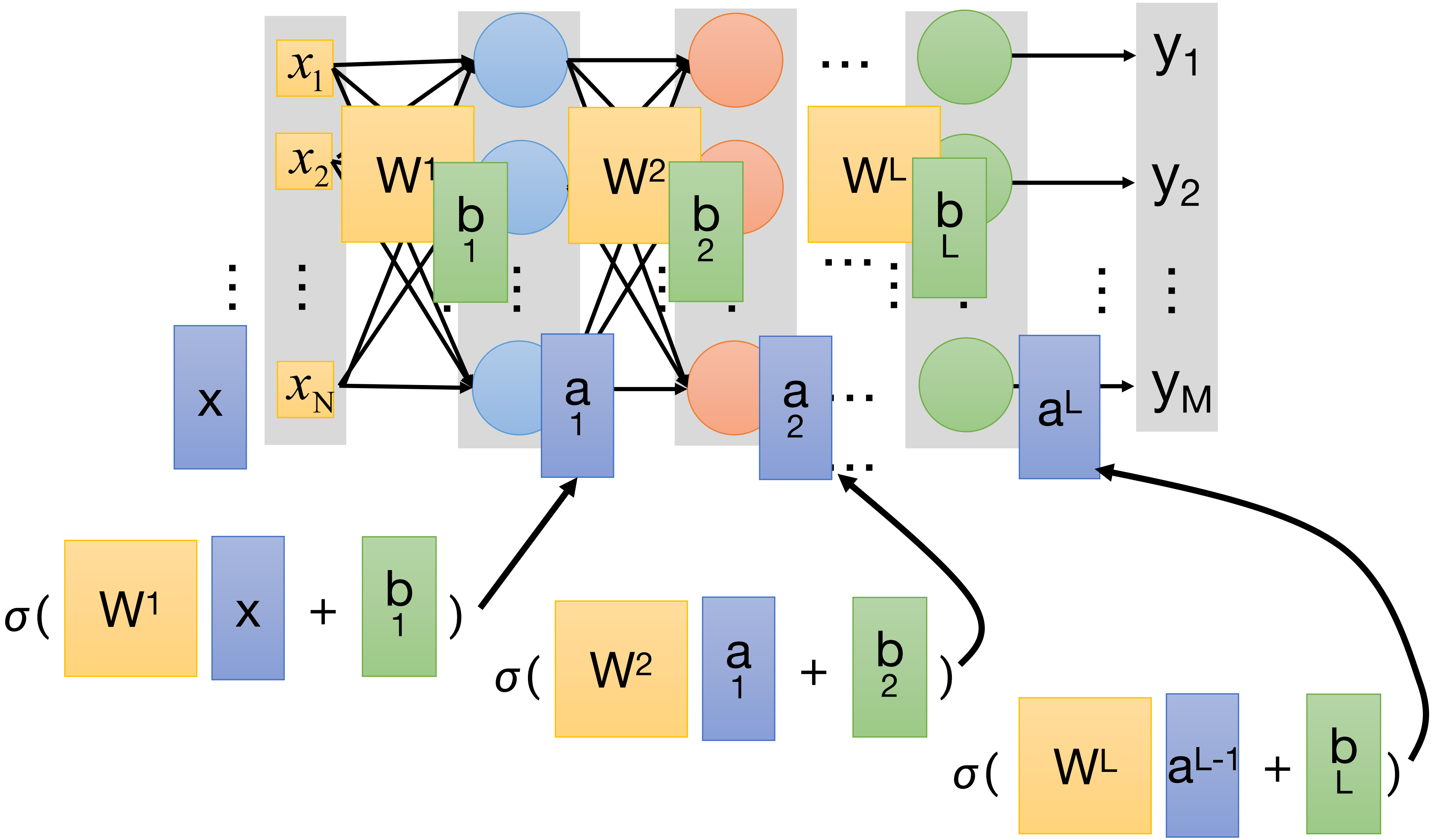
Given network structure, define ***a function set***

Matrix Operation

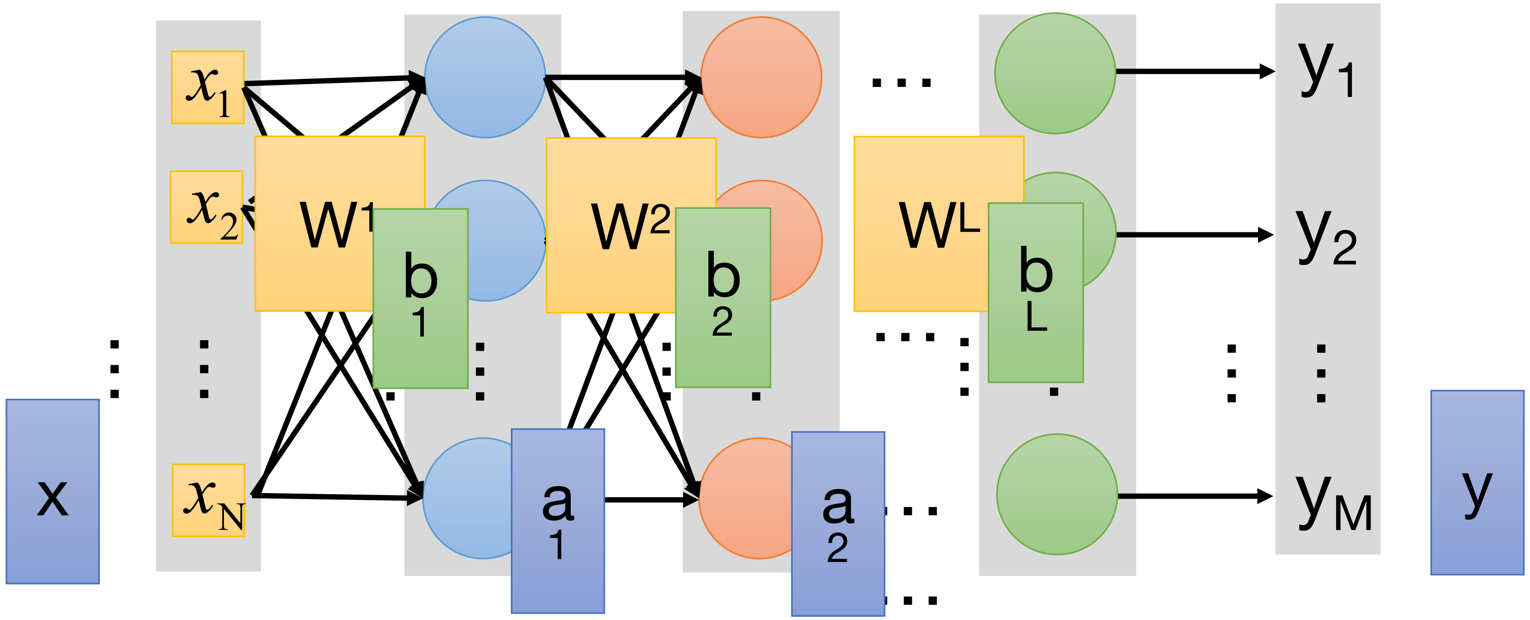


$$\sigma\Bigg(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Bigg) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



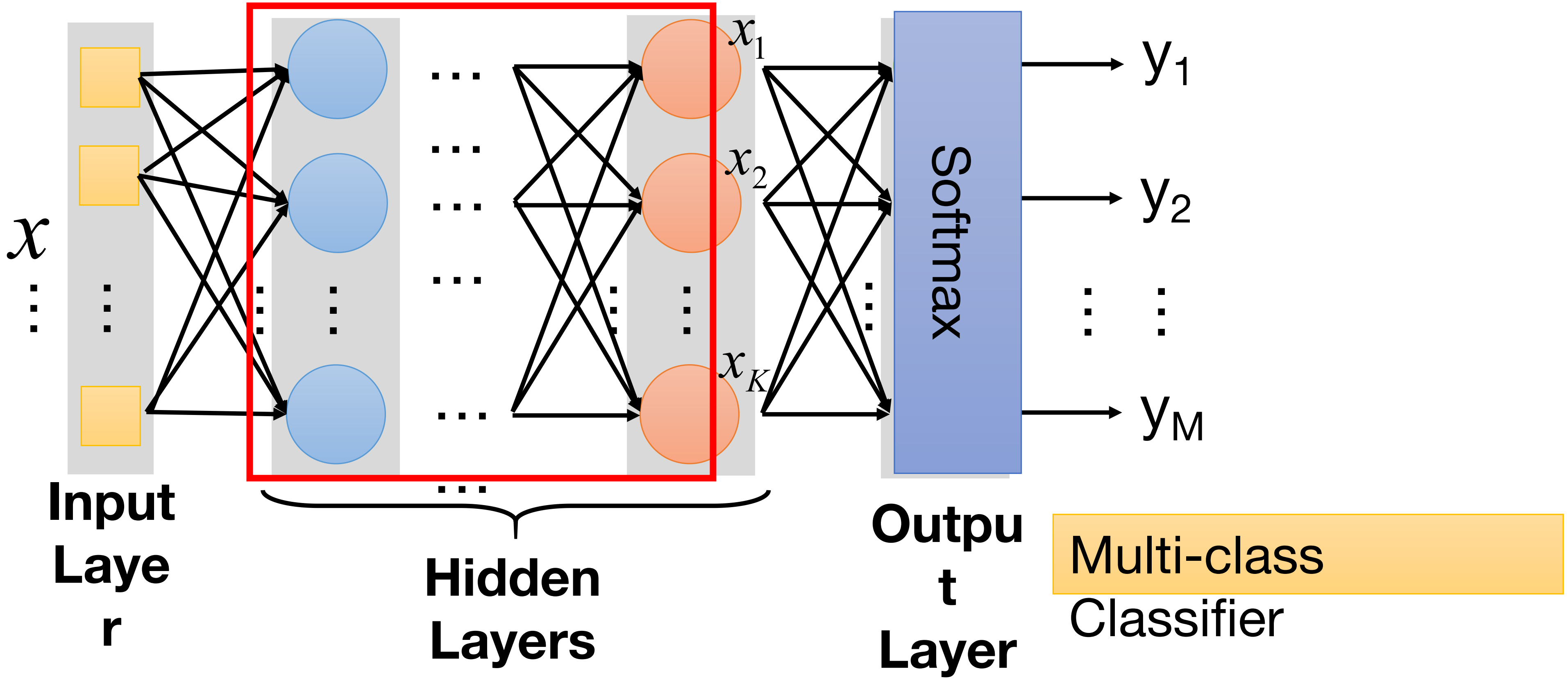
Neural Network



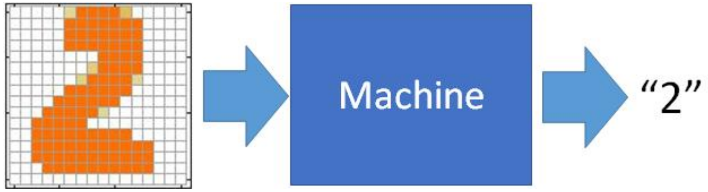
$$y = f(x)$$

$$= \sigma(W^L \cdots \sigma(W^2\, \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

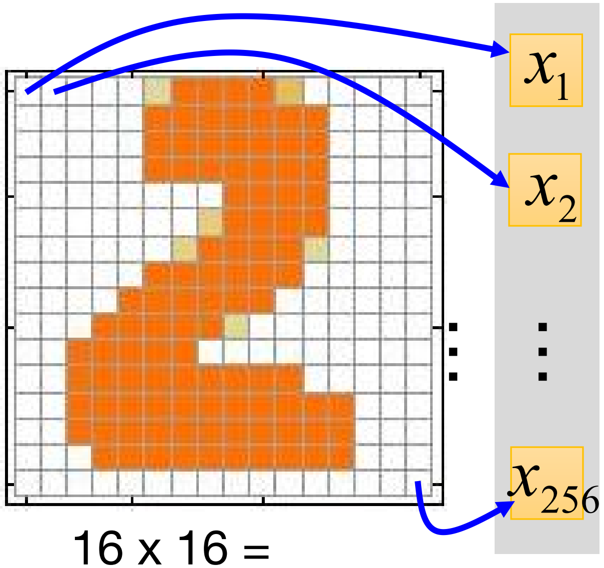
Output Layer as Multi-Class Classifier



Example Application



Input

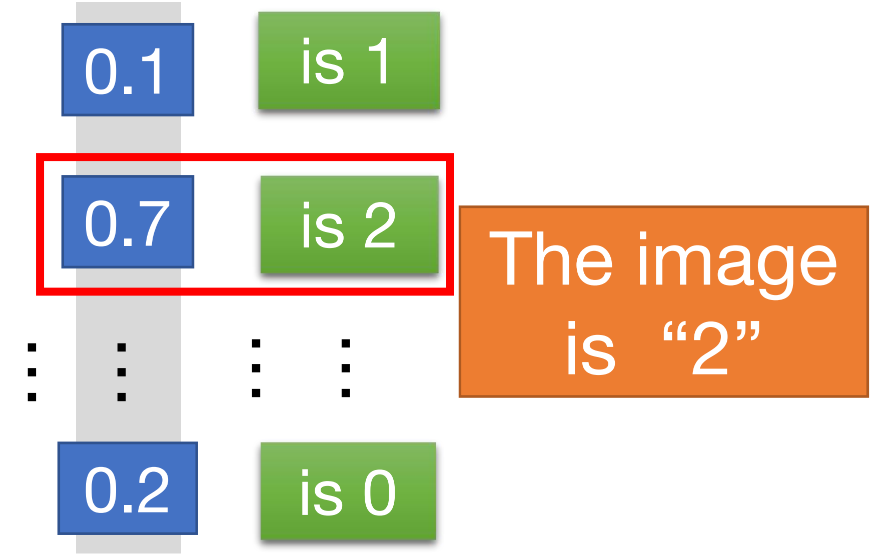


16 x 16 = 256

Ink → 1

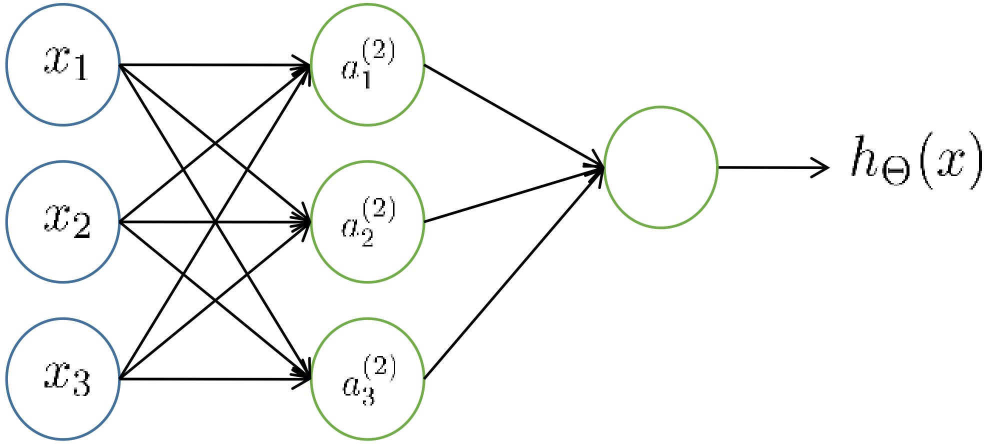
No ink → 0

Output



Each dimension represents the confidence of a digit.

- model representation



$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

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

Vectorized implementation

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

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

Add $a_0^{(2)} = 1$

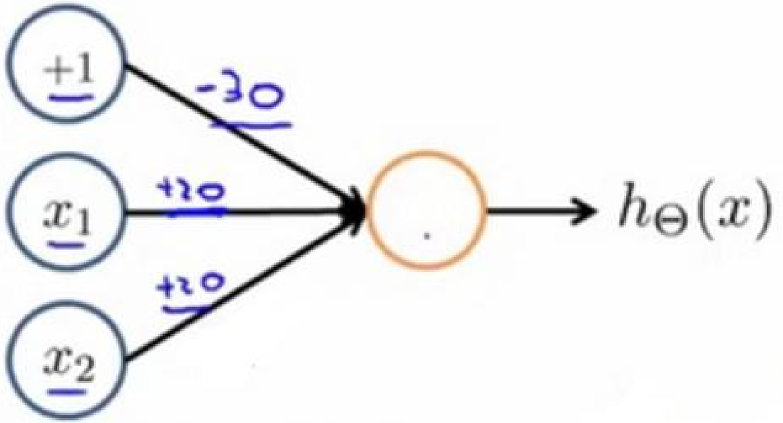
$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

- NN for logic expressions

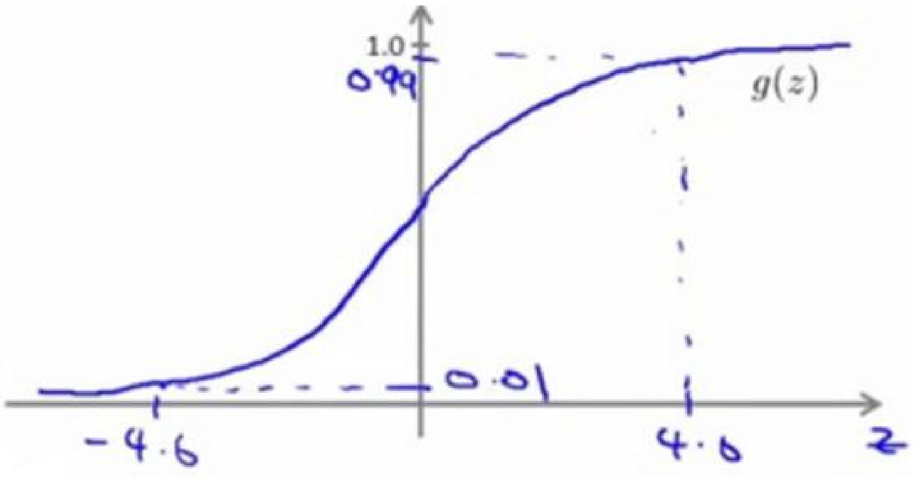
Simple example: AND

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

$y = x_1 \text{ AND } x_2$

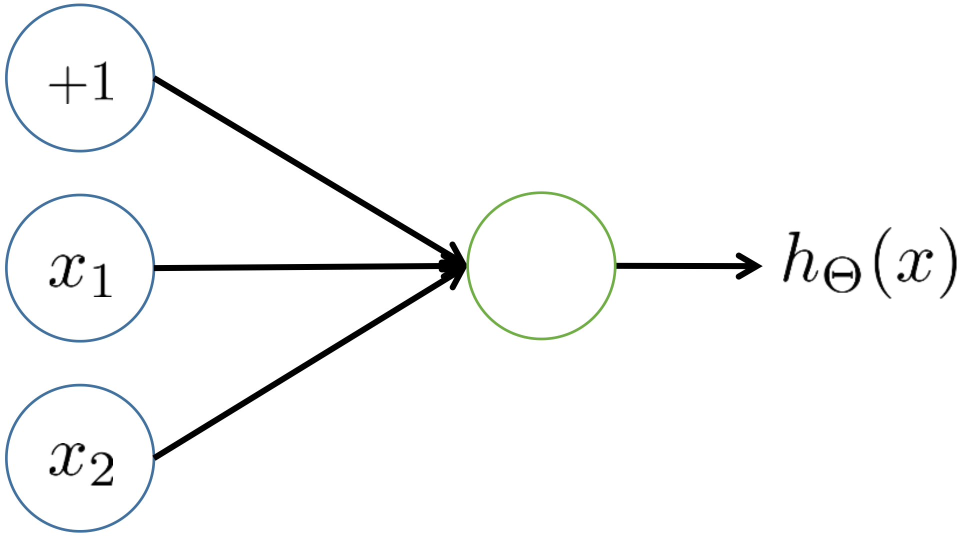


$$\rightarrow h_\Theta(x) = g(\underbrace{-30}_{\Theta^{(1)}_{10}} + \underbrace{20}_{\Theta^{(1)}_{11}} x_1 + \underbrace{20}_{\Theta^{(1)}_{12}} x_2) \leftarrow$$



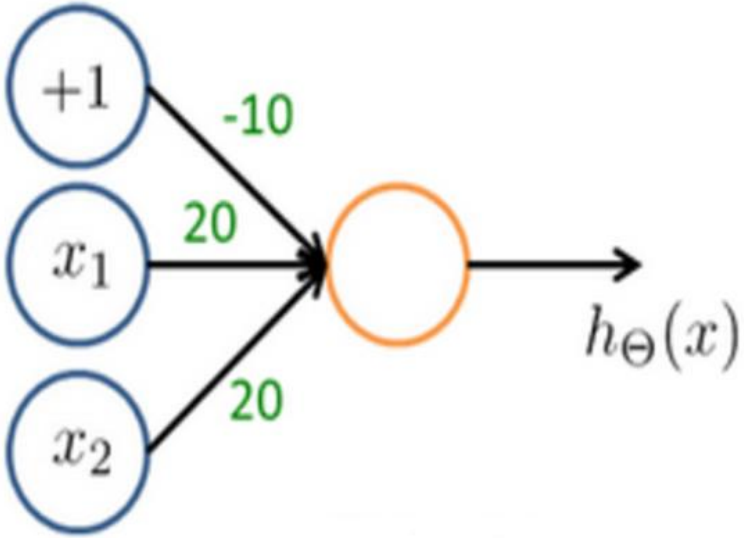
x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

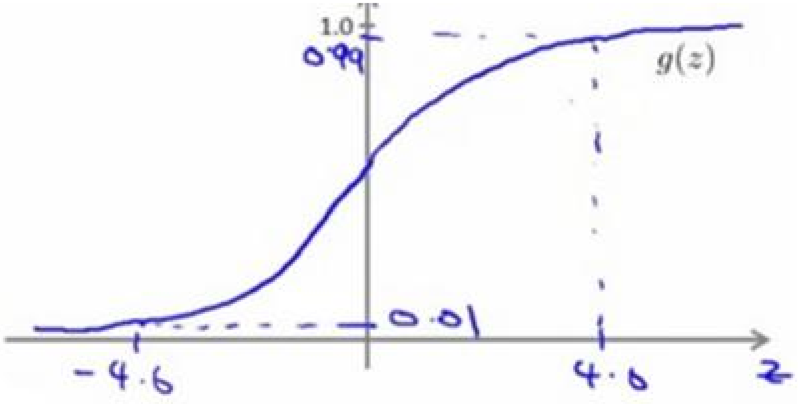
Example: OR



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

Example: OR





x1	x2	z	g(z)
0	0	-10	0
0	1	10	1
1	0	10	1
1	1	30	1

Putting it together: $x_1 \text{ XNOR } x_2$

(x1 && x2) || (¬x1) && (¬x2) = x1 XNOR x2

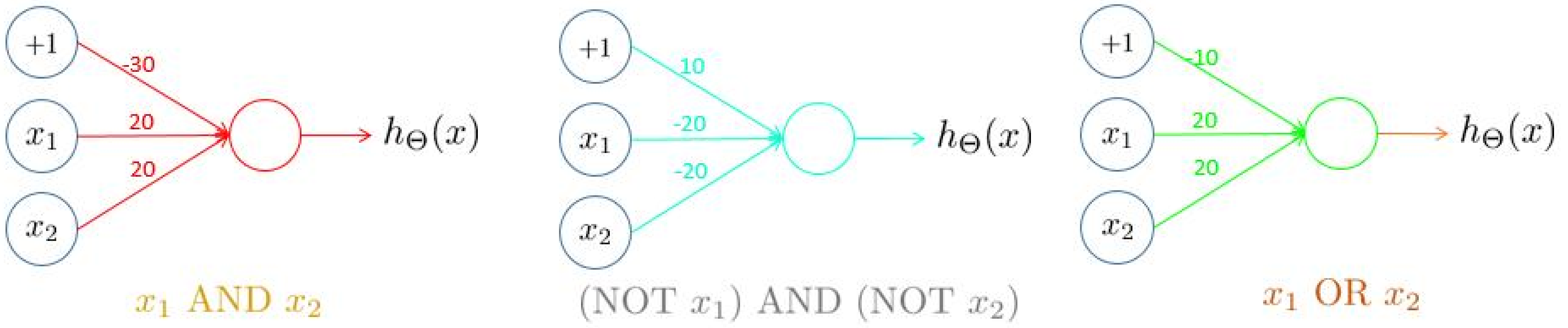
a^2_1 = x1 && x2

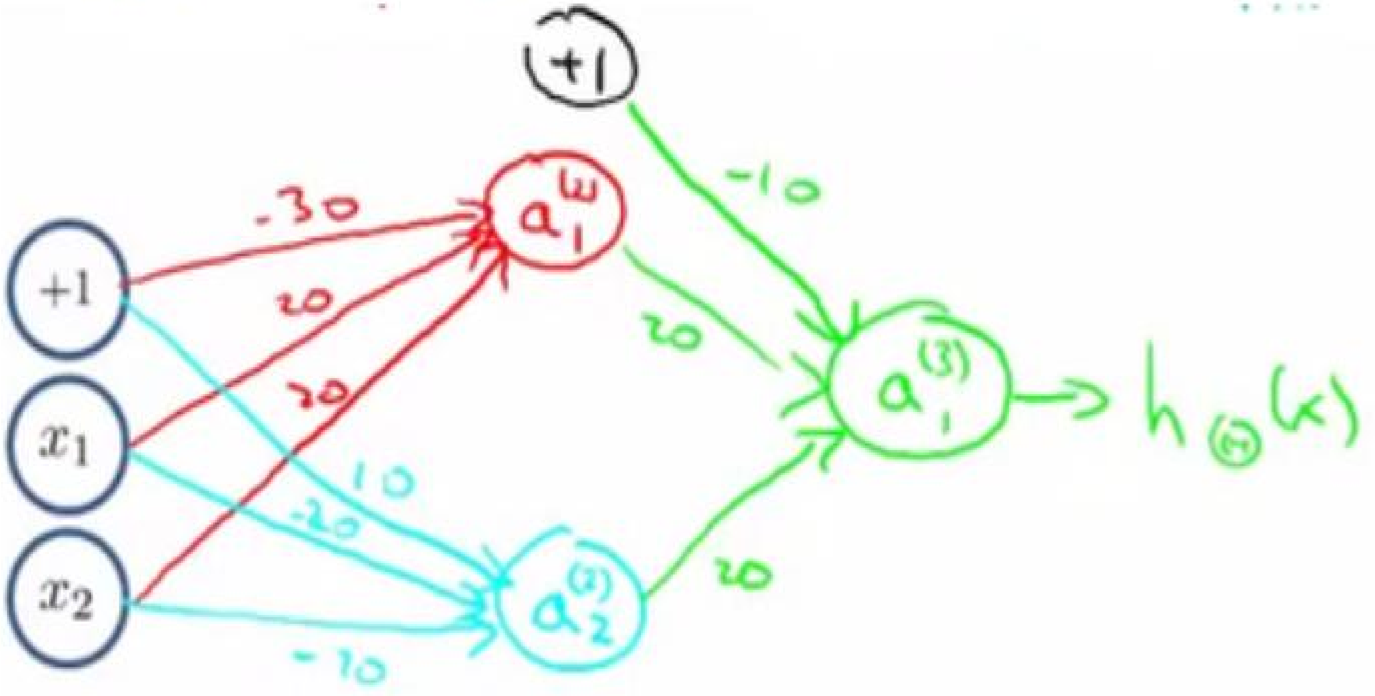
a^2_2 = (¬x1) && (¬x2)

a^3_1 = a^2_1 || a^2_2

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

$a^3_1 = a^2_1 \,||\, a^2_2$ =(x1 && x2) || (¬x1) && (¬x2) = x1 XNOR x2

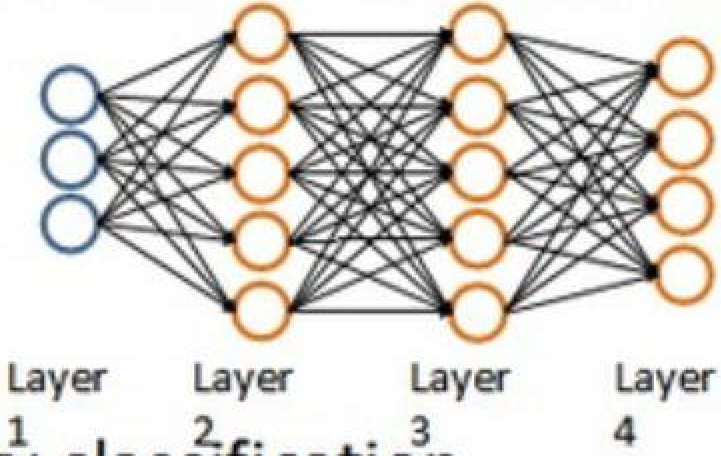




x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	C	0	0	0
1	1	1	0	1

- FCN for classification

Neural Network (Classification)



$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$L =$ total no. of layers in network

$s_l =$ no. of units (not counting bias unit) in layer l

Binary classification

$y = 0$ or 1

1 output unit

Multi-class classification (K classes)

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

K output units

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

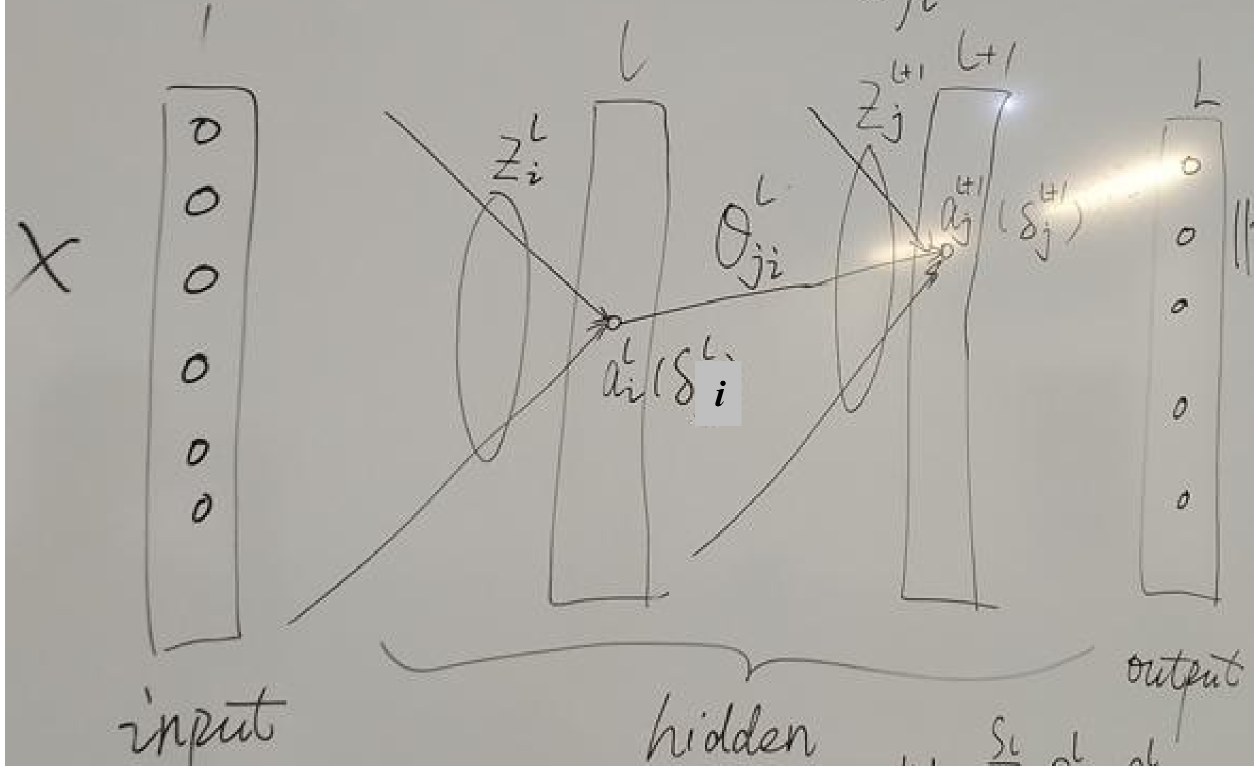
Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

farward: $a_i^l \to a_j^{l+1}$

back: $\delta_j^{l+1} \to \delta_i^l \to \theta_{ji}^l$



$$a_j^{l+1} = g(z_j^{l+1}) \qquad z_j^{l+1} = \sum_i^{S_l} \theta_{ji}^l \cdot a_i^l$$

N_l, S_l

$$a_i^l = g(z_i^l)$$

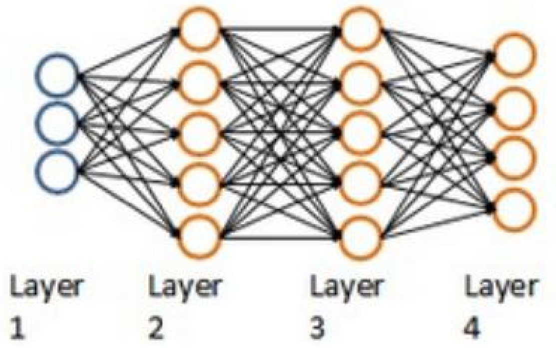
- # Backpropagation algorithm

Gradient computation

Given one training example (x, y):

Forward propagation:

$$a^{(1)} = x$$
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$
$$z^{(4)} = \Theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$



Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

- Updating weight

$$E = \frac{1}{2}\sum_i (y_i - a_i)^2 \qquad \Delta W \propto -\frac{\partial E}{\partial W} \Longrightarrow \Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} + \Delta\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha\frac{\partial E(\Theta)}{\partial\Theta_{ij}^{(l)}}$$

- Residual error for last layer (distance between a_i

$$\delta_i^{(l)} = \frac{\partial E}{\partial z_i^{(l)}} = \frac{\partial \frac{1}{2}(y - a_i^{(l)})^2}{\partial z_i^{(l)}} = \frac{\partial[\frac{1}{2}(y - g(z_i^{(l)}))^2]}{\partial z_i^{(l)}} = (a_i^{(l)} - y) \cdot g'(z_i^{(l)})$$

- Previous layers (chain rule + weighted average of all errors of next layer)

$$\delta_i^{(l)} = \frac{\partial E}{\partial z_i^{(l)}} = \sum_j^{N^{(l+1)}} \frac{\partial E}{\partial z_j^{(l+1)}} \cdot \frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}$$

$$= \sum_j^{N^{(l+1)}} \delta_j^{(l+1)} \cdot \frac{\partial[\sum_k^{N^l} \Theta_{jk}^{(l)} \cdot g(z_k^{(l)})]}{\partial z_i^{(l)}}, i \in k$$

$$= \sum_j^{N^{(l+1)}} (\delta_j^{(l+1)} \cdot \Theta_{ji}^{(l)}) \cdot g'(z_i^{(l)})$$

$$\frac{\partial E}{\partial \theta_{ji}^l} = \frac{\partial E}{\partial z_j^{(l+1)}} \cdot \frac{\partial z_j^{(l+1)}}{\partial \theta_{ji}^l} = \delta_j^{(l+1)} \cdot a_i^{(l)}$$

$$\theta_{ji}^l = \theta_{ji}^l - \alpha \cdot \frac{\partial E}{\partial \theta_{ji}^l} = \theta_{ji}^l - \alpha \cdot \delta_j^{(l+1)} \cdot a_i^{(l)}$$

Until reach minimal loss

Backpropagation algorithm

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = (a_i^{(l)} - y) \cdot g'(z_i^{(l)})$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

reset weights : $\theta_{ji}^{l} = \theta_{ji}^{l} - \alpha \cdot \delta_j^{(l+1)} \cdot a_i^{(l)}$

$$a_j^{l+1} = g\left(\sum_{i=1}^{S_l} \theta_{ji}^{l} \cdot a_i^{l} \right)$$

$$\boxed{\delta_i^{(l)} = \frac{\partial E}{\partial z_i^{(l)}} = \sum_j^{N^{(l+1)}} (\delta_j^{(l+1)} \cdot \Theta_{ji}^{(l)}) \cdot g'(z_i^{(l)})}$$