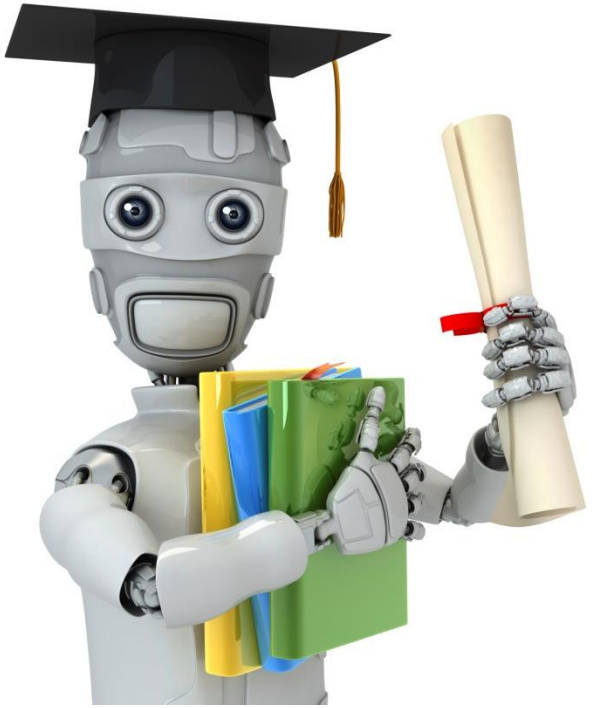
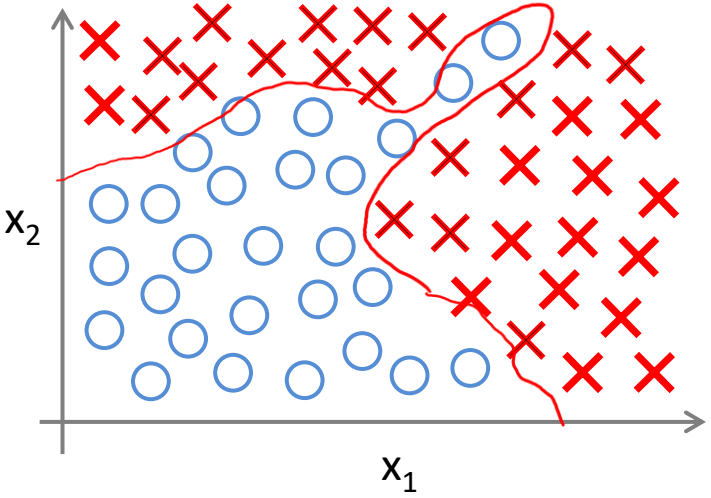


Machine Learning

Neural Networks: Representation

Non-linear hypotheses

Non-linear Classification



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

$\theta_0 + \theta_1 x_1 + \dots + \theta_{100} x_{100} + \theta_{101} x_1^2 + \theta_{102} x_1 x_2 + \theta_{103} x_1 x_3 + \theta x_2^2 + \theta x_2 x_3 + x_{100}^2$

$x_1 =$ size
$x_2 =$ # bedrooms
$x_3 =$ # floors
$x_4 =$ age
$\dots$
x_{100}

What is this?

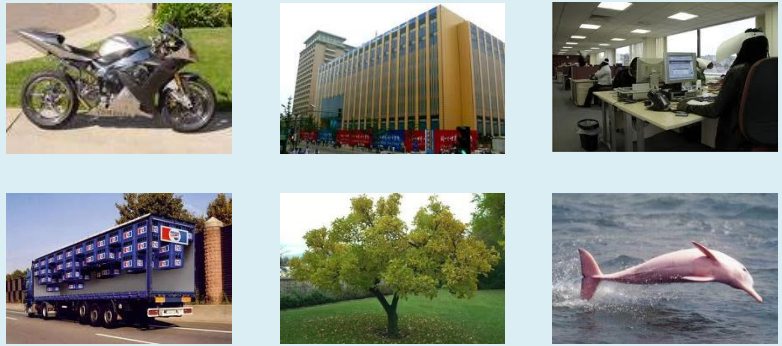
You see this:

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Computer Vision: Car detection

Cars

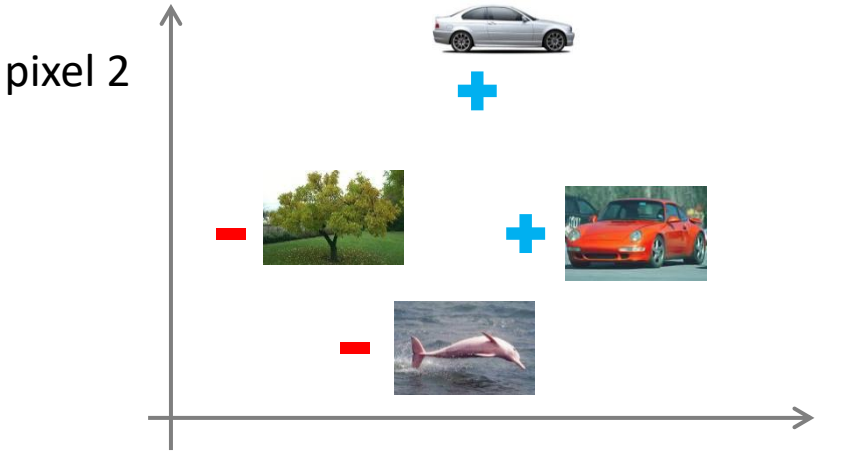


Not a car

Testing:

What is this?

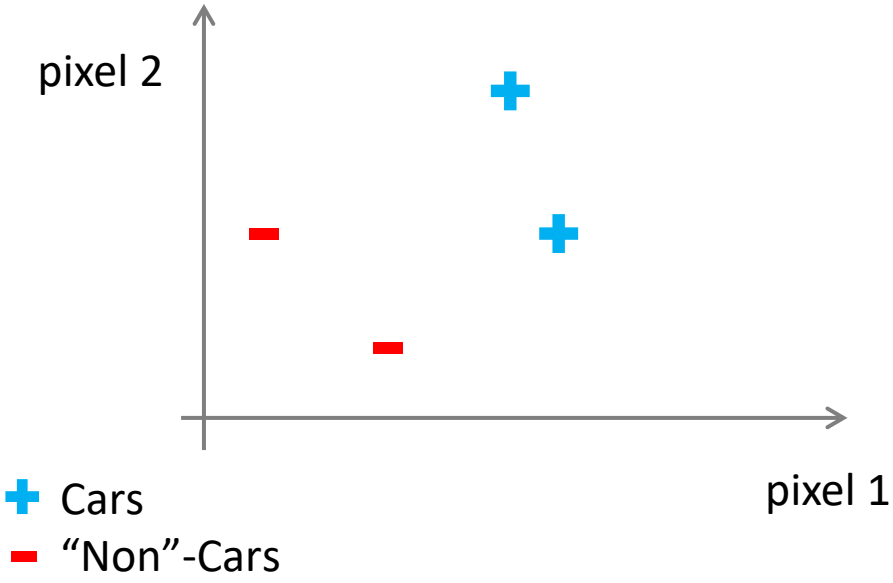
Learning Algorithm

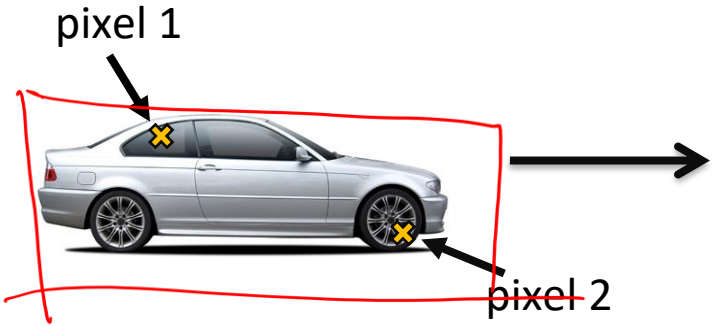


+ Cars

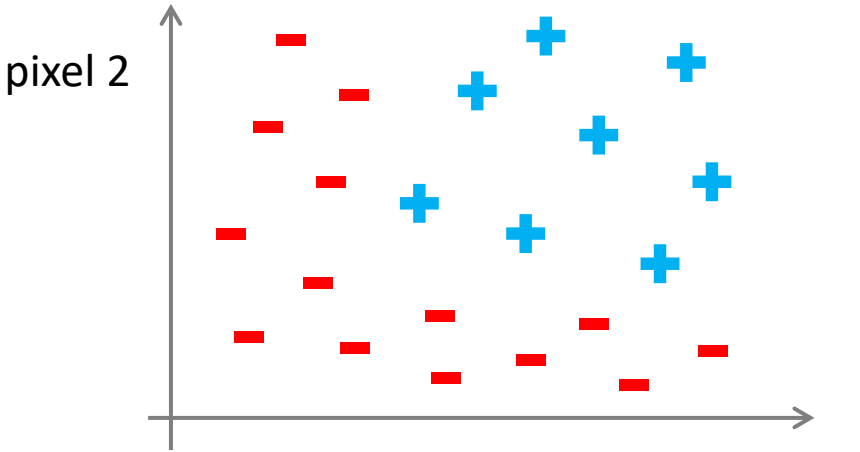
– “Non”-Cars

Learning Algorithm





Learning Algorithm



Cars

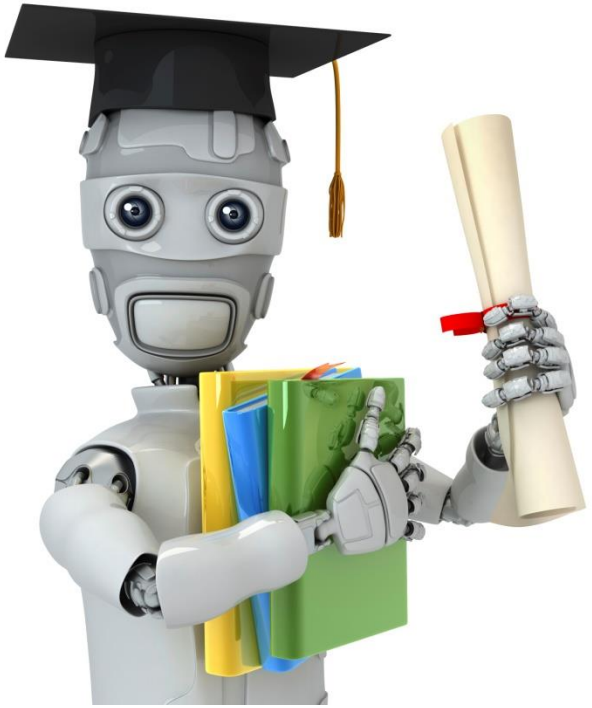
"Non"-Cars

50 x 50 pixel images→ 2500 pixels

$n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

Quadratic features ($x_i \times x_j$): ≈3 million features



Machine Learning

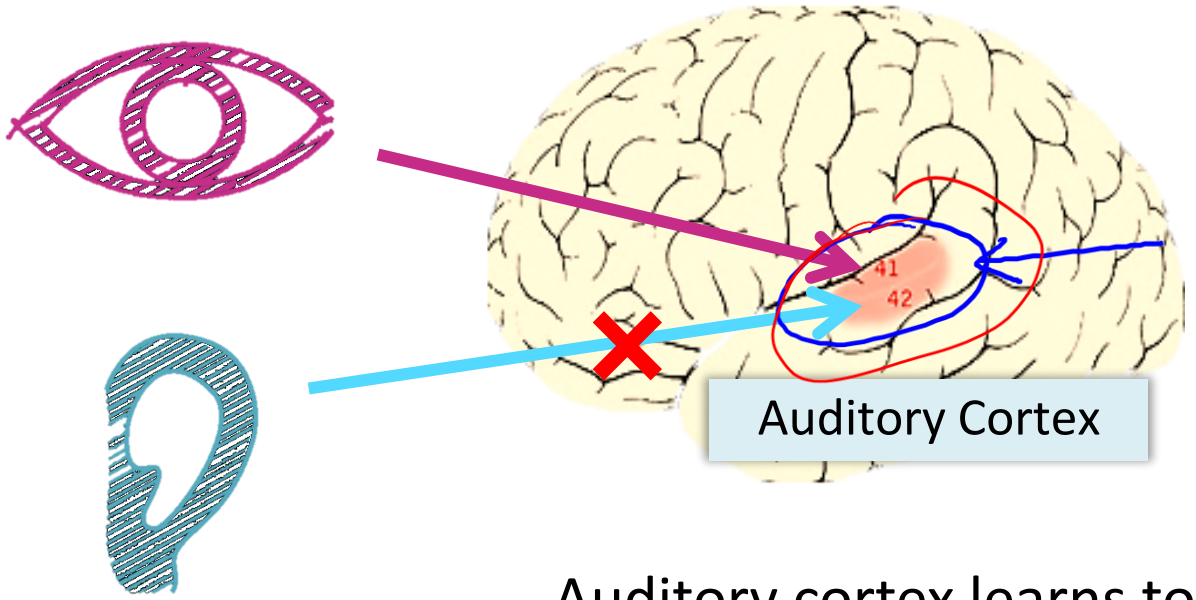
Neural Networks: Representation

Neurons and the brain

Neural Networks

- Origins: Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications

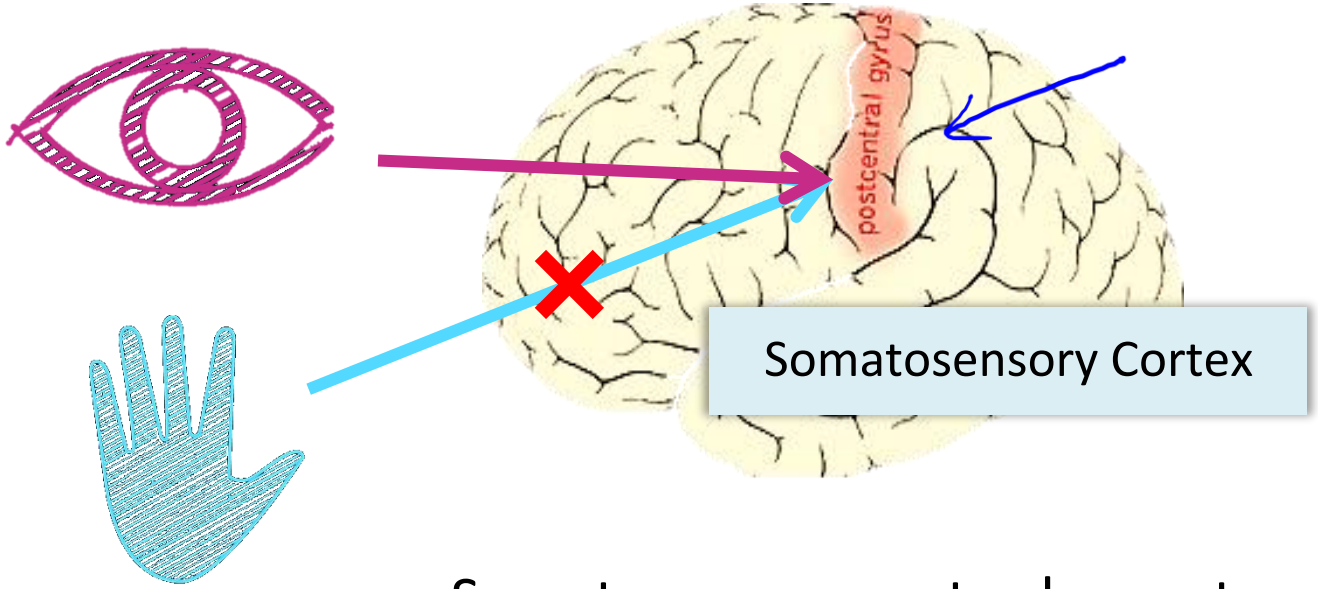
The “one learning algorithm” hypothesis



Auditory cortex learns to see

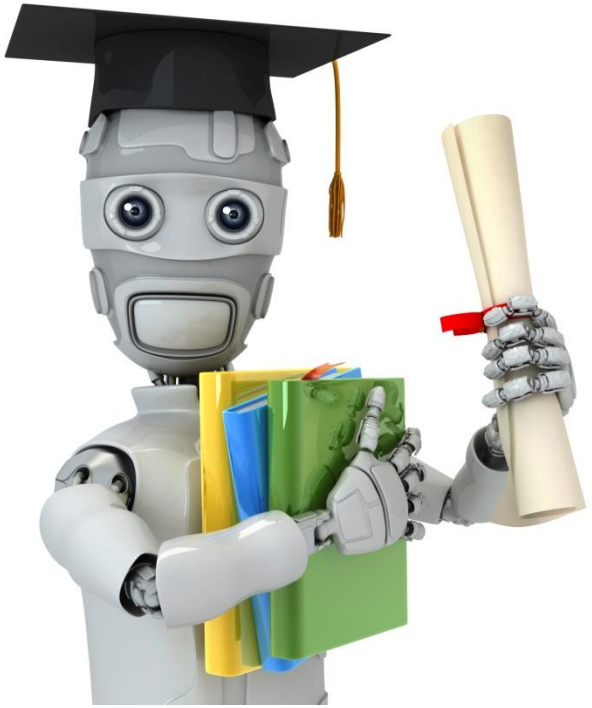
[Roe et al., 1992]

The “one learning algorithm” hypothesis



Somatosensory cortex learns to see

[Metin & Frost, 1989]

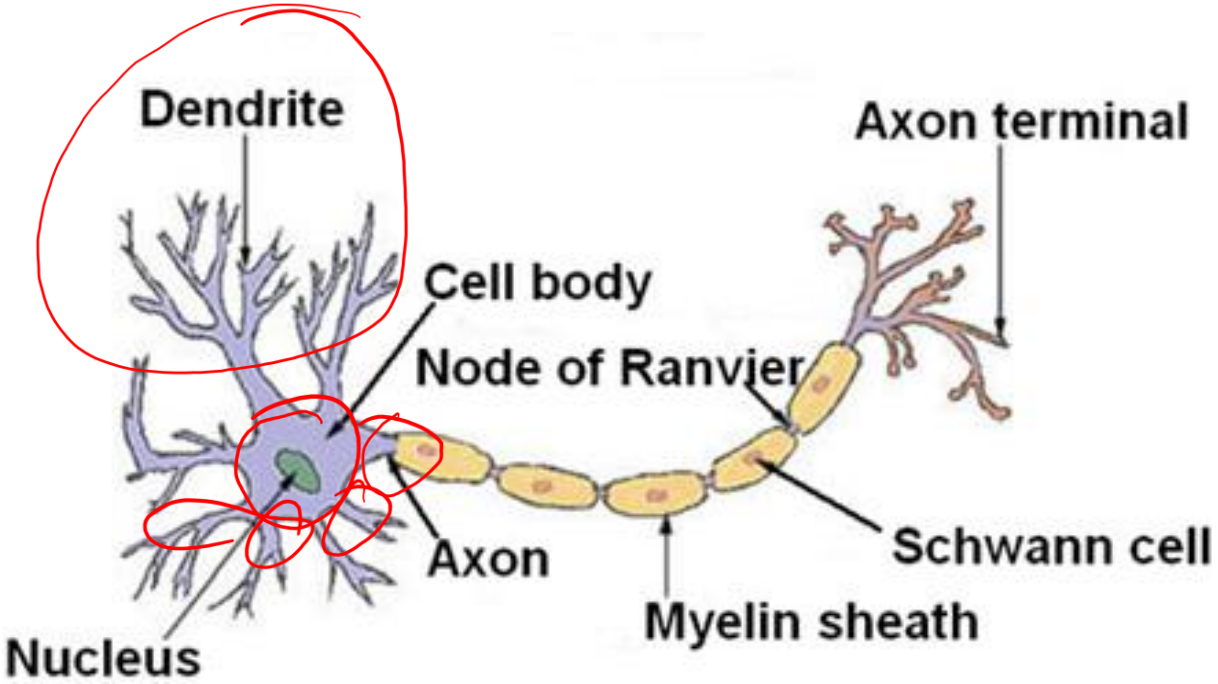


Machine Learning

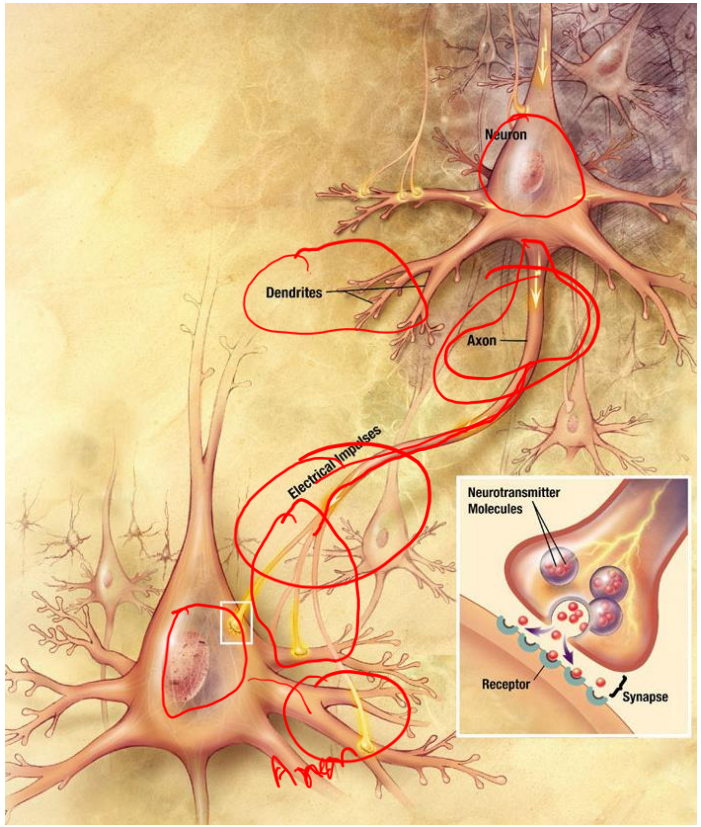
Neural Networks: Representation

Model representation I

Neuron in the brain

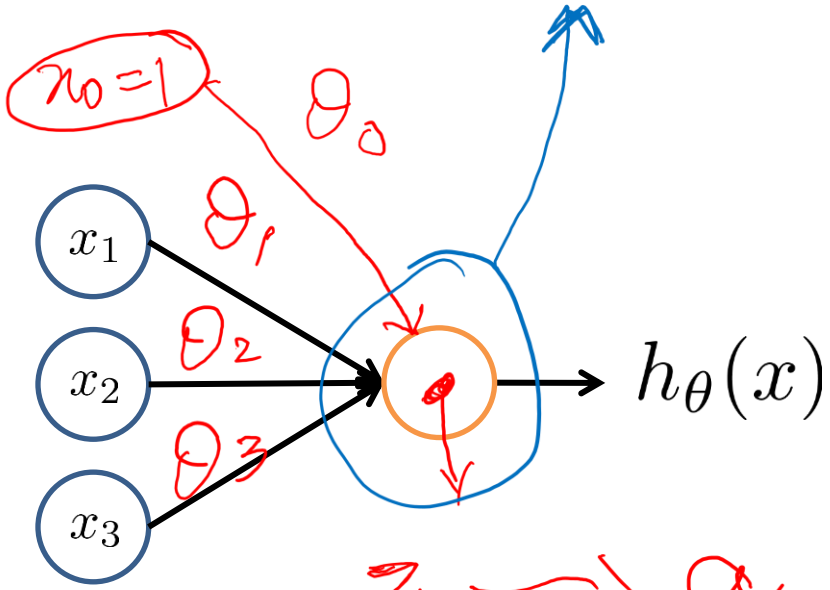


Neurons in the brain



[Credit: US National Institutes of Health, National Institute on Aging]

Neuron model: Logistic unit



$$x = \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$z \rightarrow a \qquad z = \theta^T x$

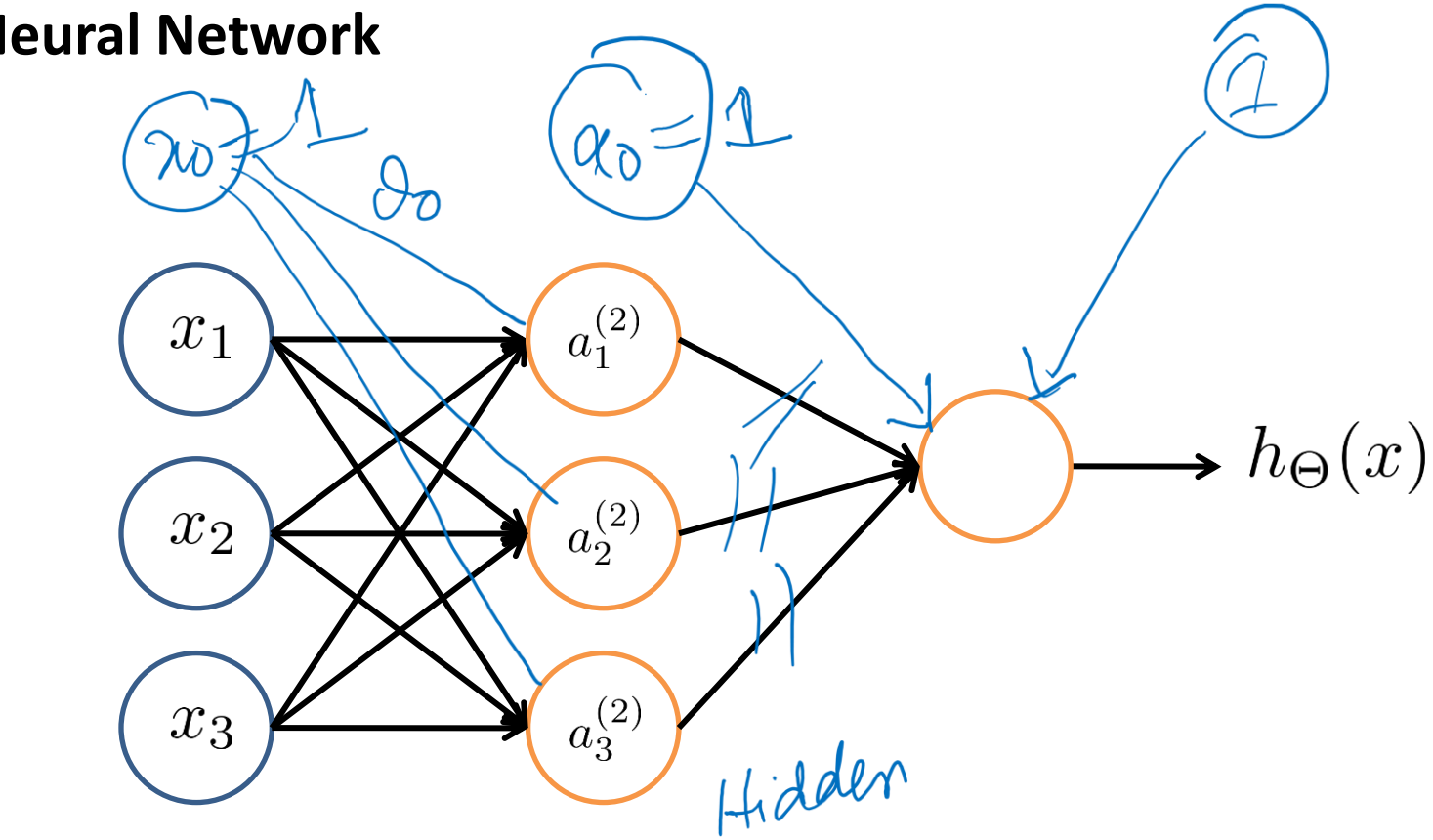
$$\frac{1}{1 + e^{-\theta^T x}}$$

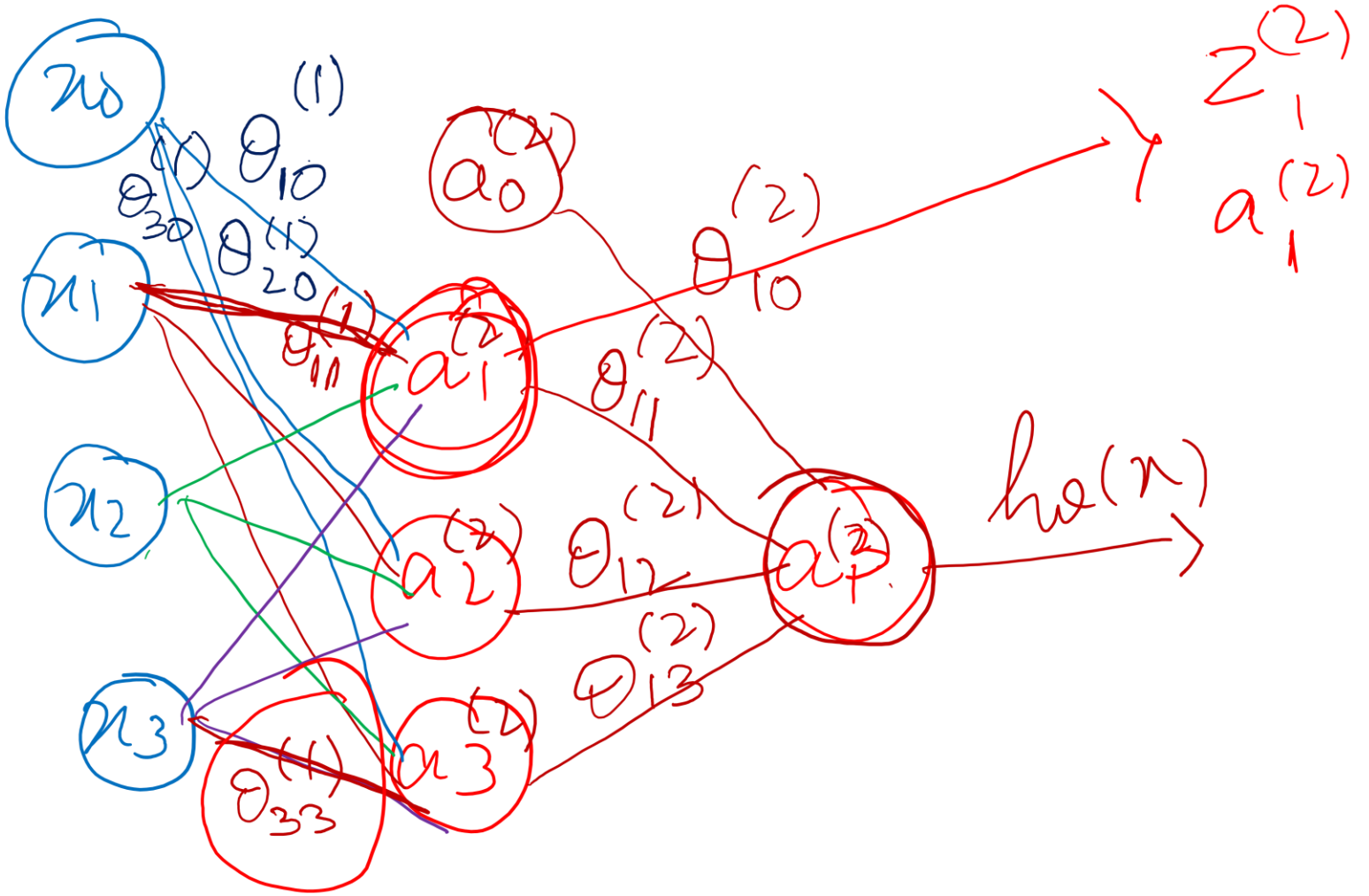
Perceptron

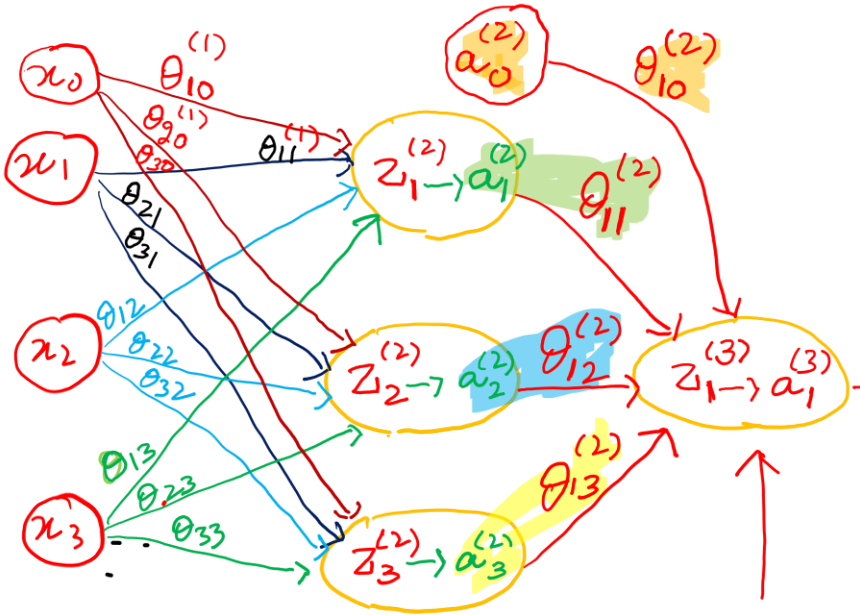
Sigmoid (logistic) activation function. $a = \sigma(z)$

Layer 1 Layer 2

Neural Network




x_0
x_1
x_2
x_3
$\theta_{10}^{(1)}$
$\theta_{30}^{(1)}$
$\theta_{20}^{(1)}$
$\theta_{11}^{(1)}$
$\theta_{33}^{(1)}$
$a_0^{(2)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_3^{(2)}$
$\theta_{10}^{(2)}$
$\theta_{11}^{(2)}$
$\theta_{12}^{(2)}$
$\theta_{13}^{(2)}$
$a^{(3)}$
$h_\theta(x)$
$z_1^{(2)}$
$a_1^{(2)}$



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

$$z_2^{(2)} = \theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3$$

$$z_3^{(3)} = \theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3$$

g: Sigmoid. σ

ReLU

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

$$a_2^{(2)} = g(z_2^{(2)})$$

$$a_3^{(2)} = g(z_3^{(2)})$$

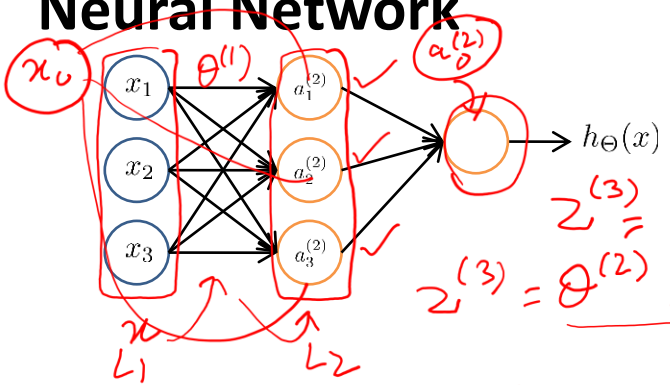
$$z_1^{(3)} = \theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)}$$

$$a_1^{(3)} = g(z_1^{(3)})$$

(i) Sigmoid.

(ii) Softmax

Neural Network

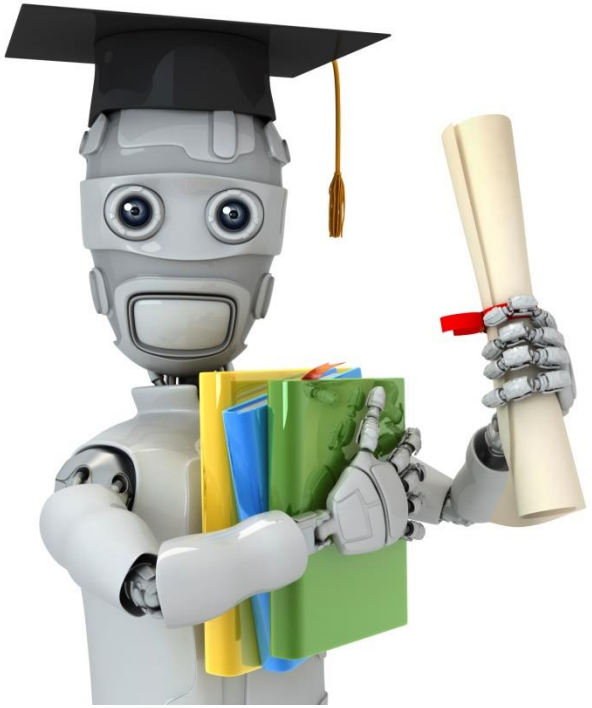


$a_i^{(j)}$ = "activation" of unit i in layer j

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

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

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

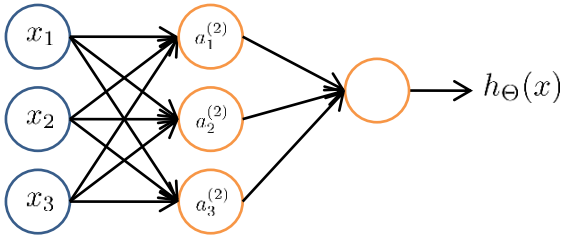


Machine Learning

Neural Networks: Representation

Model representation II

Forward propagation: Vectorized implementation



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

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

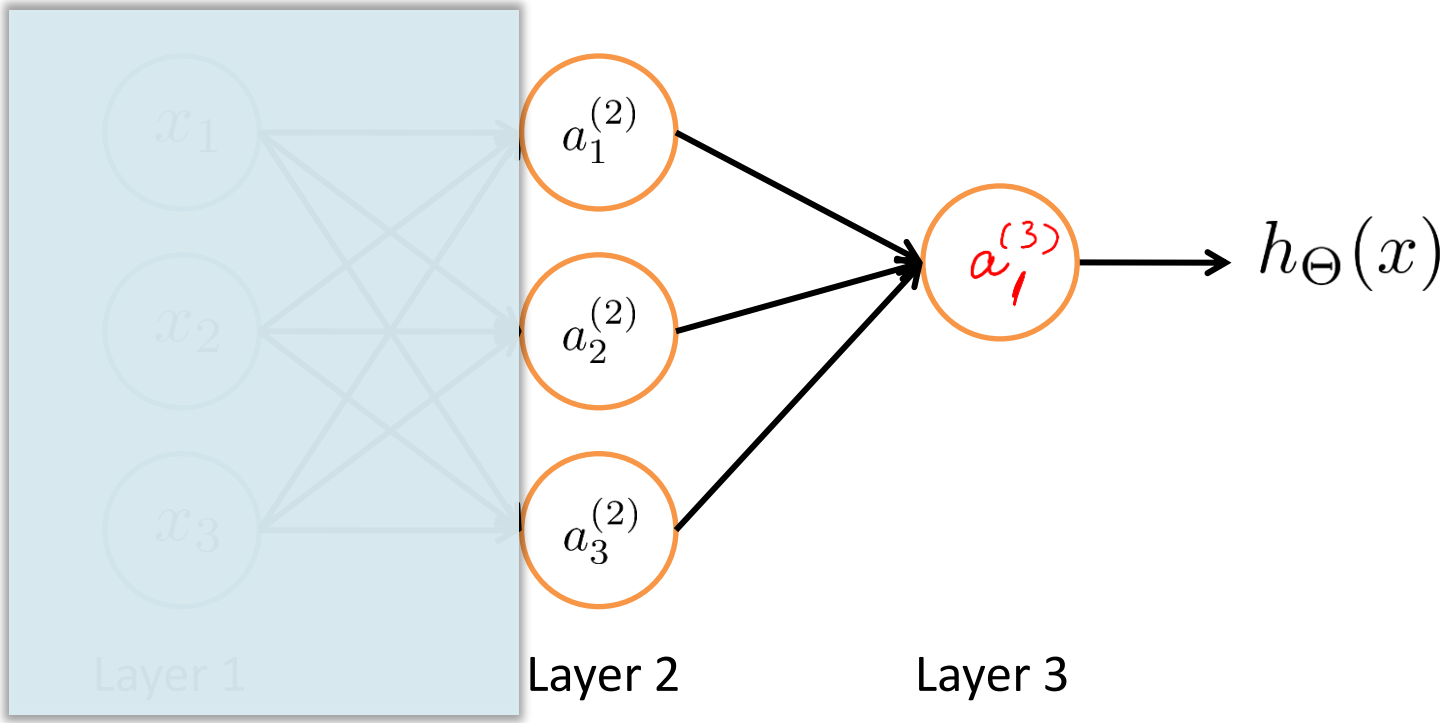
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

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

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

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

Neural Network learning its own features

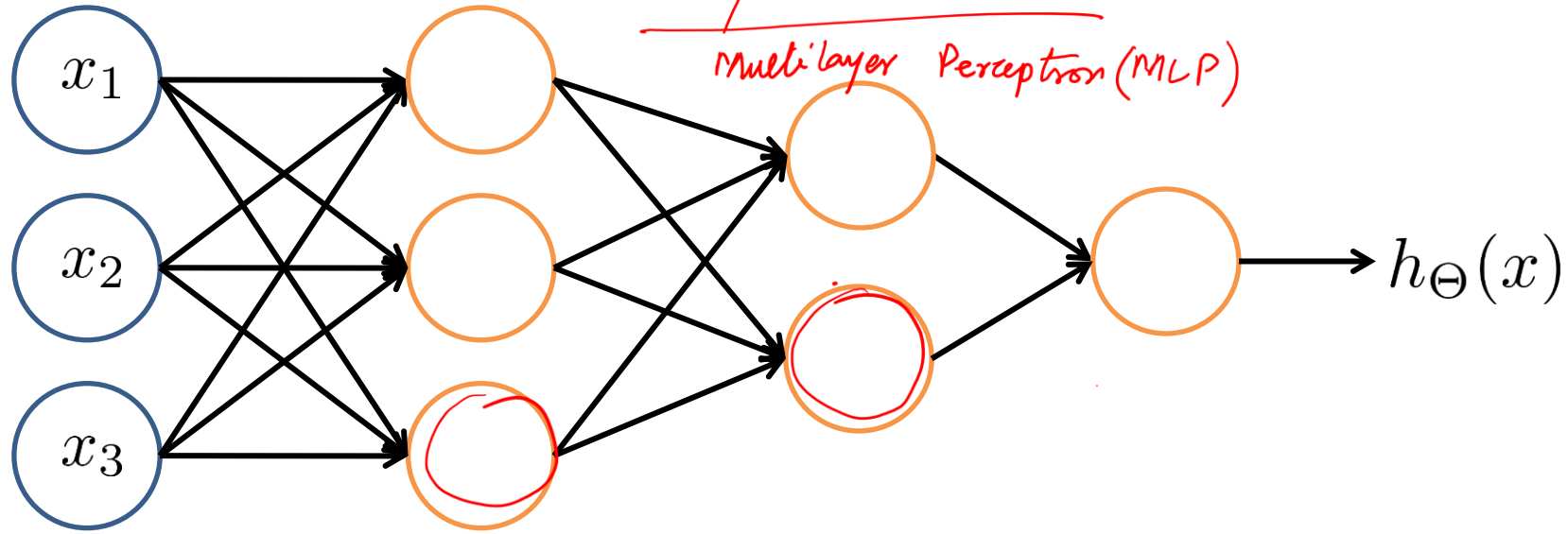


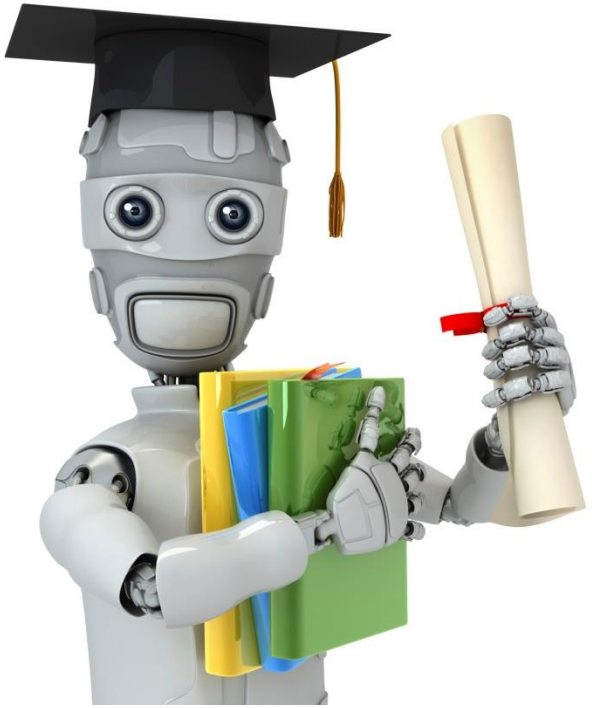
Other network architectures

(i) tensorflow Dense
(ii) Pytorch Linear

Fully connected NN

Multi layer Perceptron (MLP)





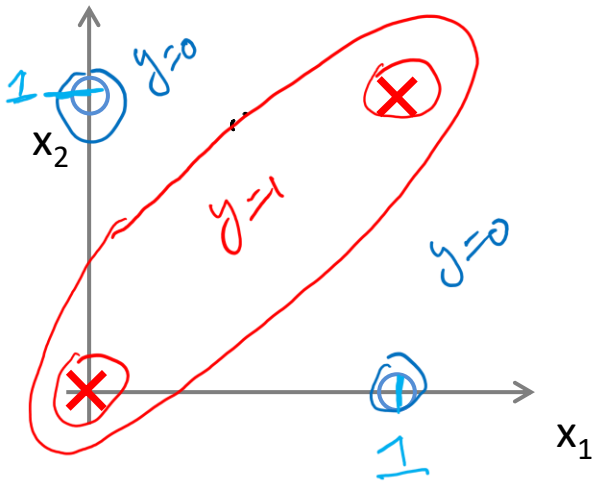
Machine Learning

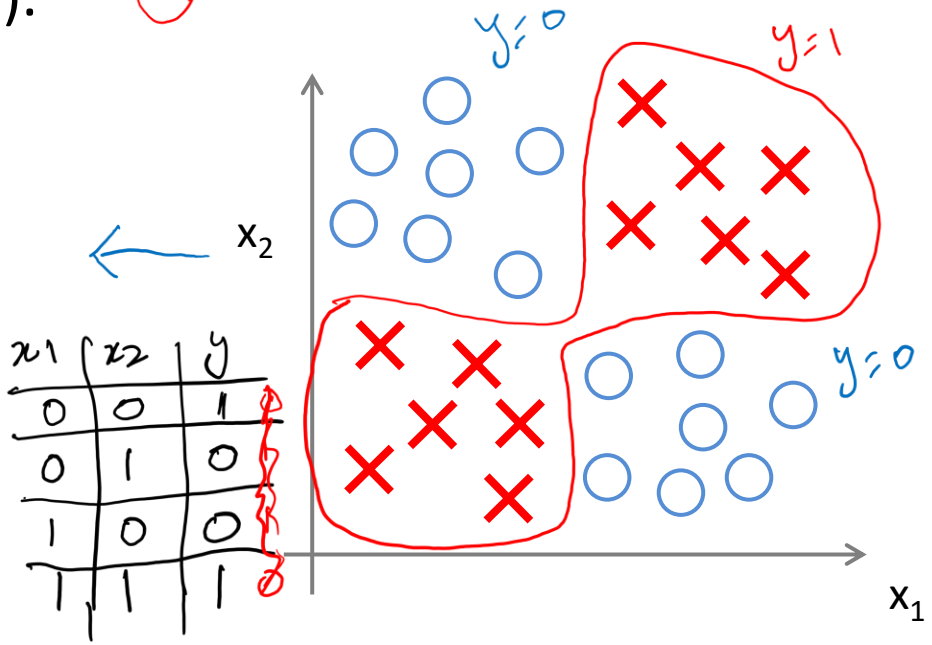
Neural Networks: Representation

Examples and intuitions I

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).





$y = x_1$ XOR x_2

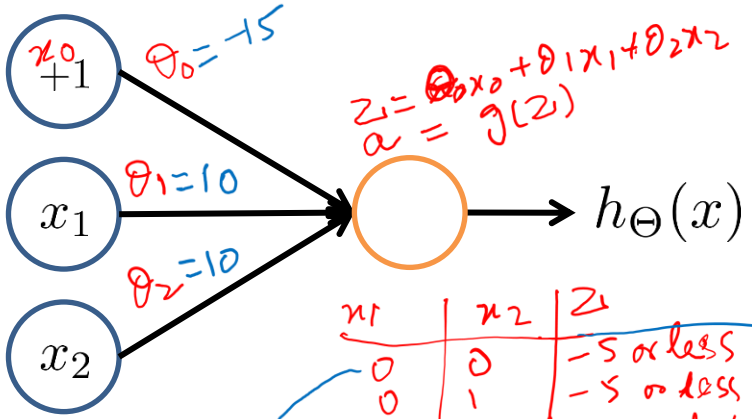
x_1 XNOR x_2

NOT $(x_1$ XOR $x_2)$

Simple example: AND

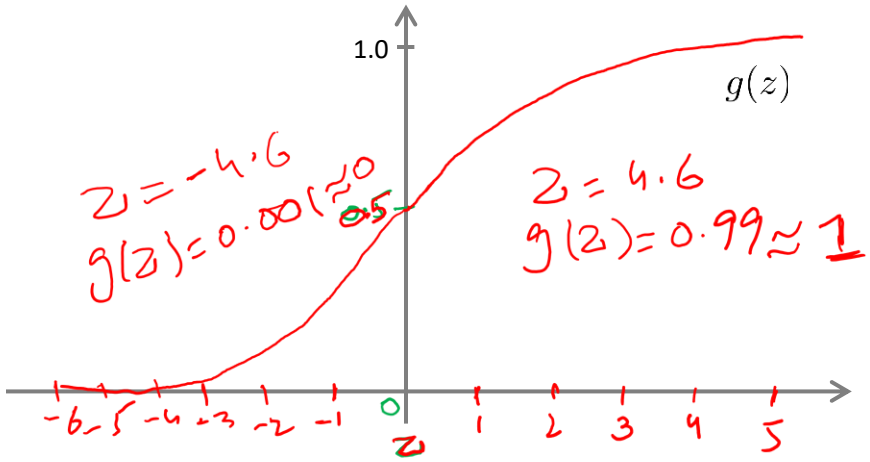
$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$



$z = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$

$a = g(z)$



x_1	x_2	z	$g(z)$
0	0	-5 or less	$g(-5) \approx 0$
0	1	-5 or less	0
1	0	-5 or less	0
1	1	$+5$ or greater	$g(+5) \approx 1$

$z = 1 \times -15 + 0 \times 10 + 0 \times 10 = -15$

$z = 1 \times -15 + 1 \times 10 + 0 \times 10 = -5$

$z = 1 \times -15 + 1 \times 10 + 1 \times 10 = +5$

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

$\sigma(z) = \frac{1}{1+e^{-z}}$

$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)$

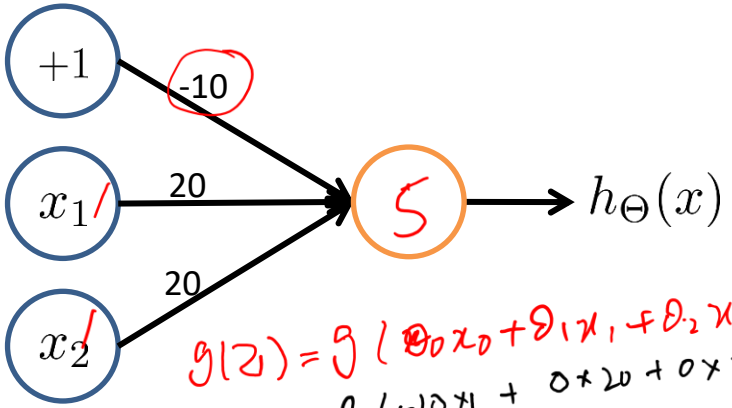
z

0 → 1

Example: OR function

$z = 4.6 \rightarrow g(z) = 0.99 \approx 1$

$z = -4.6 \rightarrow g(z) = 0.001 \approx 0$



$g(z) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)$

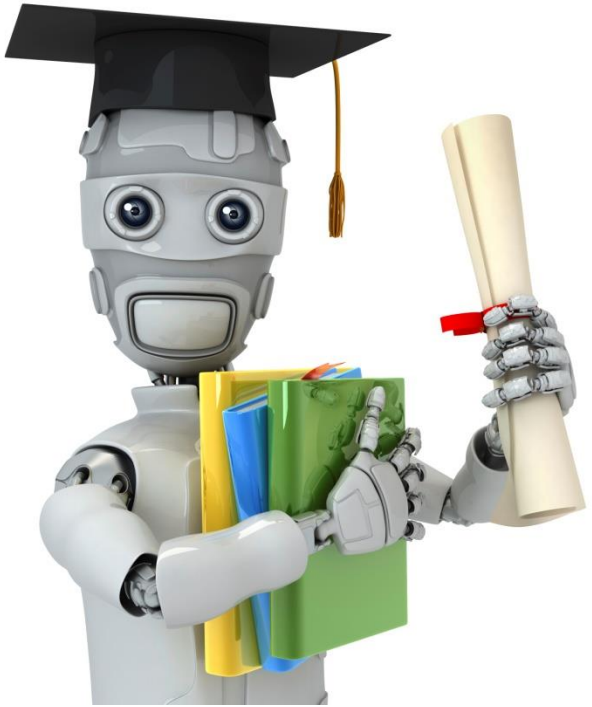
$= g(-10 \times 1 + 0 \times 20 + 0 \times 20)$

$= g(-10)$

$g(z) = g(-10 + 40) = g(30)$

$a = g(z)$

x_1	x_2	z	$h_\Theta(x)$
0	0	-5 or less	0
0	1	$+5$	1
1	0	$+5$ or greater	1
1	1	$+5$	1



Machine Learning

Neural Networks: Representation

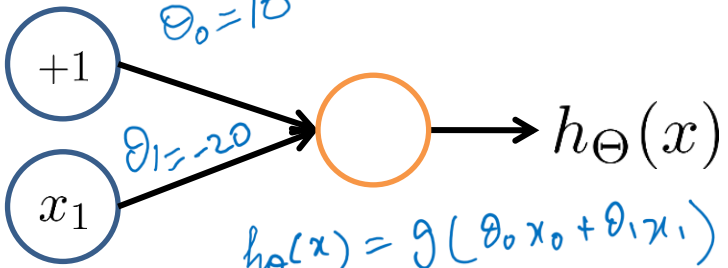
Examples and intuitions II

x_1 AND x_2

x_1 OR x_2

x_0	x_1	and	nand	and′
0	0	0	1	1
0	1	0	1	0
1	0	0	1	0
1	1	1	0	0

Negation:



$h_\Theta(x) = g(\theta_0 x_0 + \theta_1 x_1)$

$= g(10 + -20 * 0)$

$= g(10)$

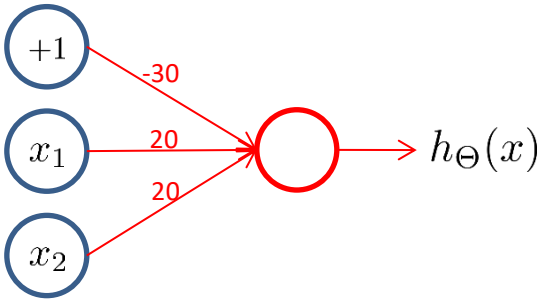
$h_\Theta(x) = g(10 - 20x_1)$

x_1	z	$h_\Theta(x)$
0	+ 5 or greater	1 ✓
1	-5 or less	0 ✓

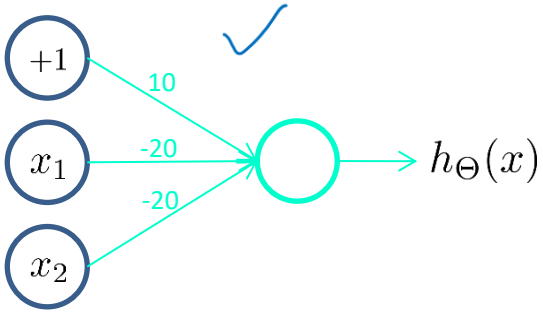
$h_\Theta(x_1) = g(10 - 20 * 1) = g(-10)$

(NOT x_1) AND (NOT x_2)

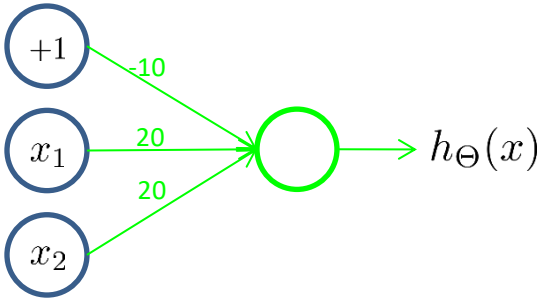
Putting it together: x_1 XNOR x_2



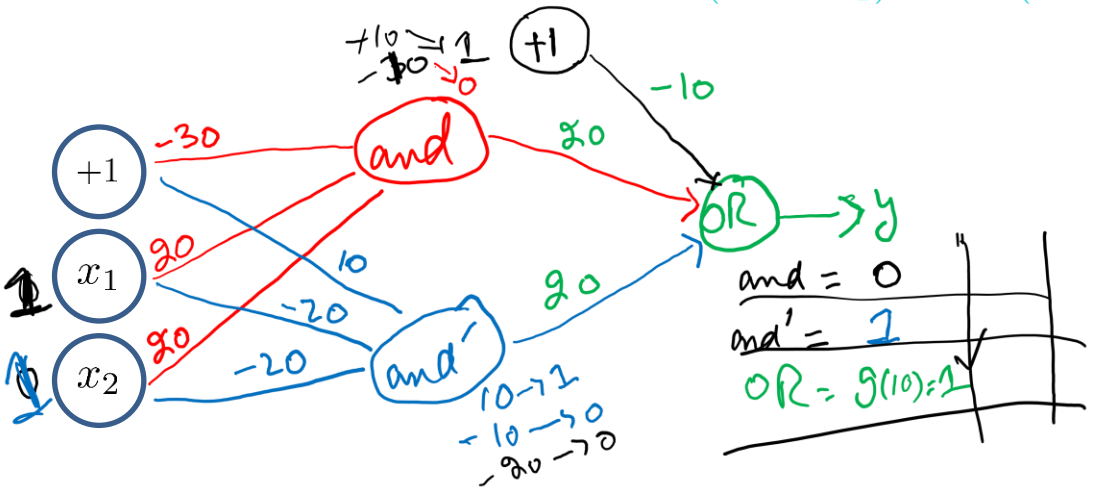
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

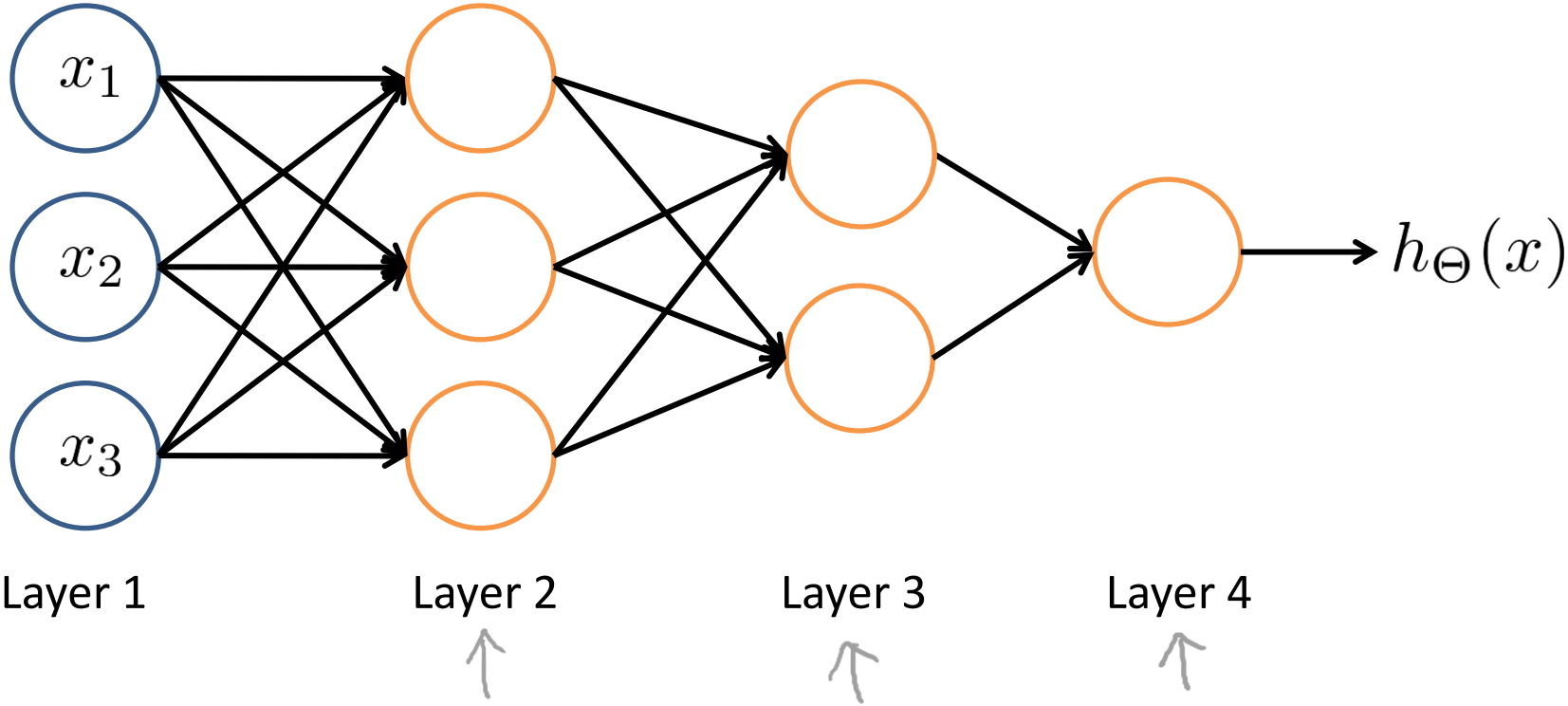


x_1 OR x_2

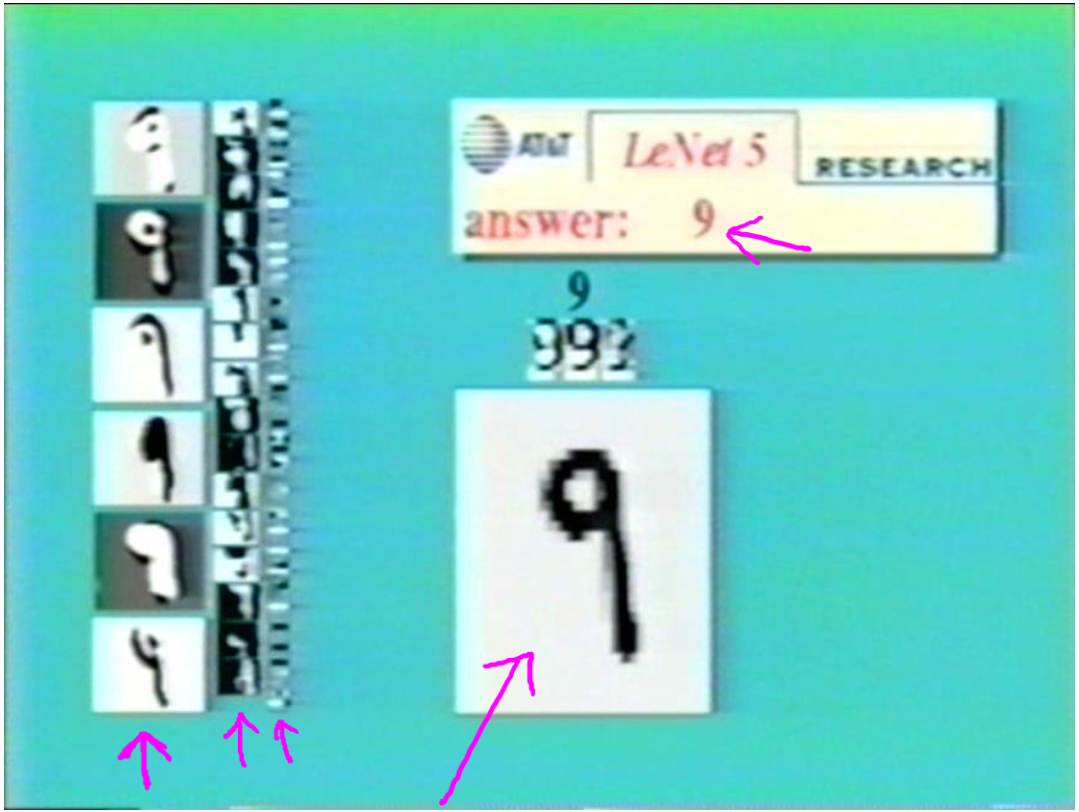


x_1	x_2	$a_1^{(2)}$ (and)	$a_2^{(2)}$ (and')	$h_\Theta(x)$ (OR)
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

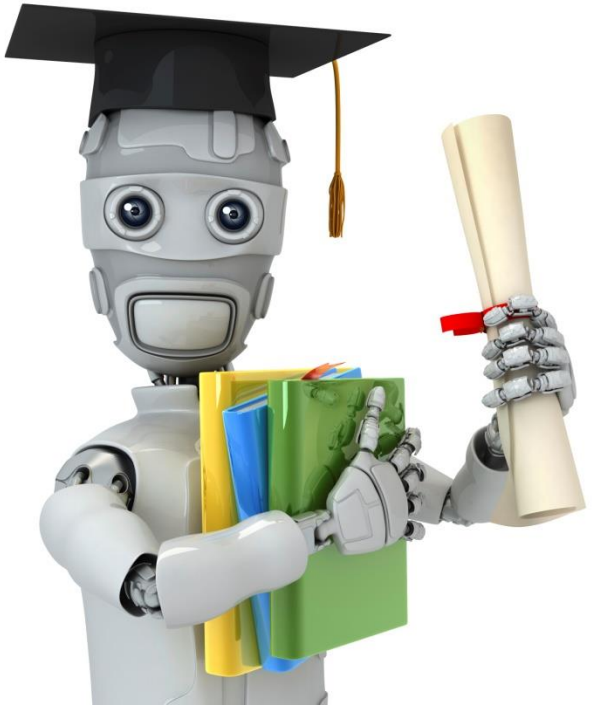
Neural Network intuition



Handwritten digit classification



[Courtesy of Yann LeCun]



Machine Learning

Neural Networks: Representation

Multi-class classification

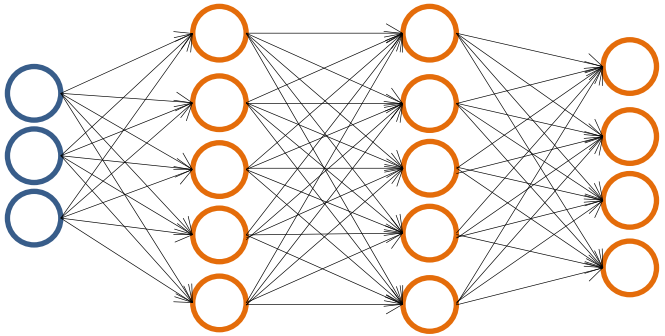
Multiple output units: One-vs-all.

Jeep

Truck

Light

Pedestrian

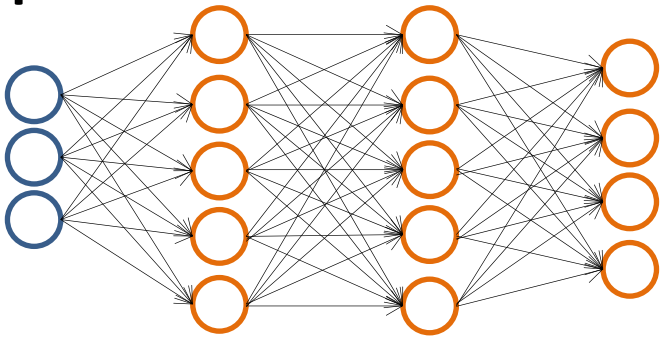


$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when jeep when truck when light

Multiple output units: One-vs-all.



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

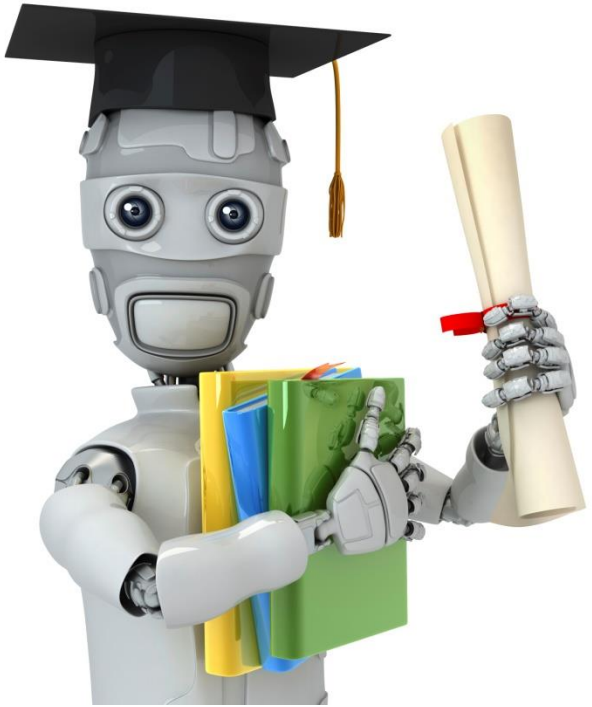
Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when Jeep when truck when light

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

$y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Jeep truck light pedestrian

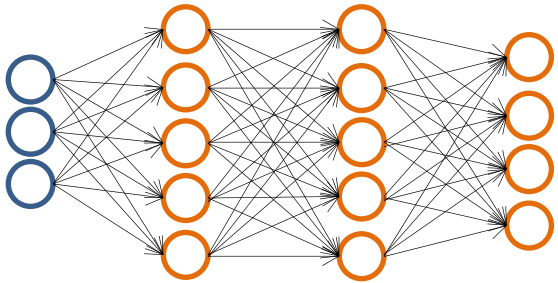


Machine Learning

Neural Networks: Learning

Cost function

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

$y = 0 \text{ or } 1$

1 output unit

Multi-class classification (K classes)

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

pedestrian car motorcycle truck

K output units

Cost function

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:

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

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log((h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Neural Network:

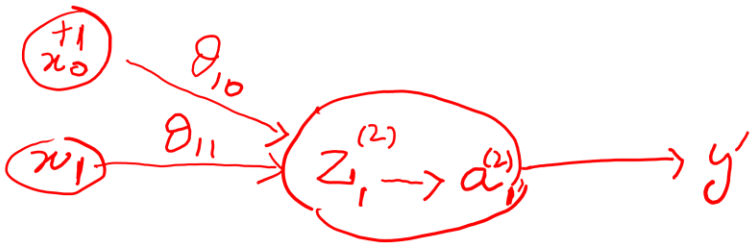
1- Forward Propagation (Predict a label)

2- cost computation (Finds the distance between actual and Predicted label)

3- Backward Propagation (optimizes the parameters and finds the ~~correct lal~~ predicted label as close as correct label)

- learning / training
  - by finding the optimal values of the parameters.
  - we use an optimizer.
    - > Gradient Descent Algorithm

1. Forward Propagation:



$$z_1^{(2)} = \theta_{10} x_0 + \theta_{11} x_1$$

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

$$y' = a_1^{(2)}$$

g is an activation function

(2) cost computation

(a) Binary cross Entropy

$$J(\theta) = -[y \log(a_1^{(2)}) + (1-y) \log(1 - a_1^{(2)})]$$

(b) Squared Error function

$$J(\theta) = (a_1^{(2)} - y)^2$$

(3) Backward / Back Propagation :

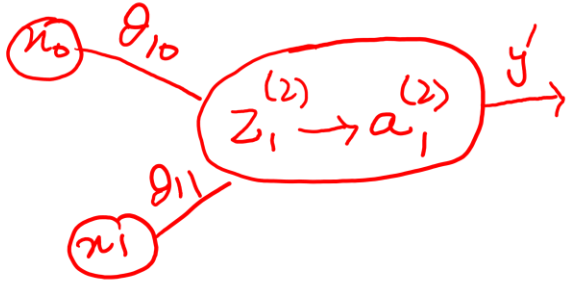
(i) Find cost by using either
   (a) Binary Cross Entropy
   or
   (b) Squared Error

(ii) use an optimezer. (GD)

θ_{10}, θ_{11}

$$\theta_{10} := \theta_{10} - \alpha \frac{\partial J(\theta)}{\partial \theta_{10}}$$

$$\theta_{11} := \theta_{11} - \alpha \frac{\partial J(\theta)}{\partial \theta_{11}}$$



Chain Rule:

if $y = f(u)$

Chain Rule

if $y = f(u)$ and

$u = g(x)$

then

$y = f(g(x))$

and

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

(i) $J(\theta) = f(a_1^{(2)}) \rightarrow f$: Binary cross Entropy / Squared Error

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

(iii) $z_1^{(2)} = \theta_{10} x_0 + \theta_{11} x_1$

Therefore,

$$\frac{\partial J(\theta)}{\partial \theta_{10}} = \frac{\partial J(\theta)}{\partial a_1^{(2)}} \; \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \; \frac{\partial z_1^{(2)}}{\partial \theta_{10}}$$

(A) (B) (C)

(A) ; In case of binary cross Entropy function;

$$\frac{\partial J(\theta)}{\partial a_1^{(2)}} = \frac{\partial}{\partial a_1^{(2)}}\left[-y \log(a_1^{(2)}) - (1-y)\log(1-a_1^{(2)})\right]$$

$$= -y \frac{\partial}{\partial a_1^{(2)}} \log(a_1^{(2)}) - (1-y)\frac{\partial}{\partial a_1^{(2)}} \log(1-a_1^{(2)})$$

$$= \frac{-y}{a_1^{(2)}} \times 1 - \frac{(1-y)}{1-a_1^{(2)}} \frac{\partial (1-a_1^{(2)})}{\partial a_1^{(2)}}$$

$$= \frac{-y}{a_1^{(2)}} - \frac{(1-y)}{1-a_1^{(2)}} * -1$$

$$= (a_1^{(2)} - y)$$

(A): In case of Squared Error function:

$$\frac{\partial J(\theta)}{\partial a_1^{(2)}} = \frac{\partial}{\partial a_1^{(2)}} \left(a_1^{(2)} - y\right)^2$$

$$= 2\left(a_1^{(2)} - y\right) \frac{\partial}{\partial a_1^{(2)}} \left(a_1^{(2)} - y\right)$$

$$= 2\left(a_1^{(2)} - y\right)$$

(B)

$$\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial}{\partial z_1^{(2)}} g(z_1^{(2)})$$

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

$$a_1^{(2)} = \sigma(z_1^{(2)})$$

$$\sigma' = \sigma(1-\sigma)$$

$$\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial}{\partial z_1^{(2)}} \sigma(z_1^{(2)})$$

$$= \sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)}))$$

$$\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = a_1^{(2)}(1 - a_1^{(2)})$$

$z_1^{(2)} = \theta_{10} x_0 + \theta_{11} x_1$

ⓒ :

$$\frac{\partial z_1^{(2)}}{\partial \theta_{10}} = \frac{\partial (\theta_{10} x_0 + \theta_{11} x_1)}{\partial \theta_{10}}$$

$$\boxed{\frac{\partial z_1^{(2)}}{\partial \theta_{10}} = x_0 = 1}$$

So, we combine Ⓐ, Ⓑ, ⓒ

$$\frac{\partial J(\theta)}{\partial \theta_{10}} = \left[\frac{-y}{a_1^{(2)}} + \frac{(1-y)}{1-a^{(2)}}\right] \times \left[a_1^{(2)}(1-a_1^{(2)})\right] \times [1]$$

$$= -y(1-a_1^{(2)}) + (1-y)\, a_1^{(2)}$$

$$= \boxed{}$$

$$\frac{\partial J(\theta)}{\partial \theta_{11}} = ?$$

$$\frac{\partial J(\theta)}{\partial \theta_{11}} = (a_1^{(2)} - y)\, x_1$$

$$\frac{\partial J(\theta)}{\partial \theta_{10}} = (a_1^{(2)} - y)$$

epochs:

- $\theta_{10} := \theta_{10} - \alpha\,(a_1^{(2)} - y)$
- $\theta_{11} := \theta_{11} - \alpha\,(a_1^{(2)} - y)\, x_1$

Forward Propagation: $a_1^{(2)} = \sigma(\theta_{10} x_0 + \theta_{11} x_1)$

if you have ~~$x^{(1)}$~~,

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

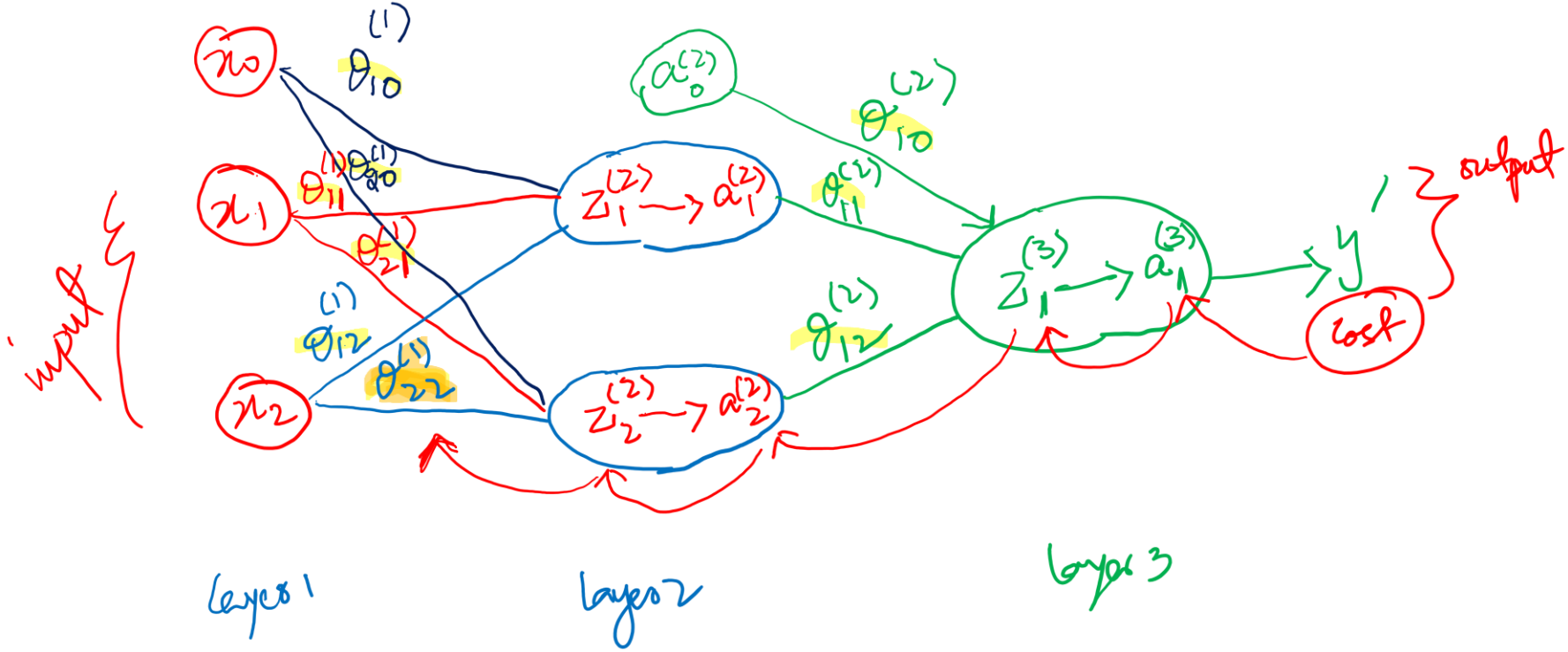
m examples.

~~A~~ B Gradient Computation:

After m examples: Batch Gradient Descent.

After $k < m$ examples: mini-batch

After each examples: Stochastic gradient Descent


x_0
x_1
x_2
input
$\theta_{10}^{(1)}$
$\theta_{11}^{(1)}$
$\theta_{20}^{(1)}$
$\theta_{21}^{(1)}$
$\theta_{12}^{(1)}$
$\theta_{22}^{(1)}$
$z_1^{(2)} \to a_1^{(2)}$
$z_2^{(2)} \to a_2^{(2)}$
$a_0^{(2)}$
$\theta_{10}^{(2)}$
$\theta_{11}^{(2)}$
$\theta_{12}^{(2)}$
$z_1^{(3)} \to a_1^{(3)}$
y
output
cost
layer 1
layer 2
layer 3

Let's compute gradients for $\theta_{22}^{(1)}$:

chain Rule.

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(1)}} = \frac{\partial J(\theta)}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial \theta_{22}^{(1)}}$$

(5) (4) (3) (2) (1)

(1) $$\frac{\partial z_1^{(2)}}{\partial \theta_{22}^{(1)}} = \frac{\partial}{\partial \theta_{22}^{(1)}} \left(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 \right)$$

$$= x_2$$

② $$\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial}{\partial z_1^{(2)}}\left(\frac{1}{1+e^{-z_1^{(2)}}}\right)$$
$$= a_1^{(2)}\left(1 - a_1^{(2)}\right)$$

③ $$\frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} = \frac{\partial}{\partial a_1^{(2)}}\left(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)}\right)$$
$$= \theta_{11}^{(2)}$$

④ $$\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = a_1^{(3)}\left(1 - a_1^{(3)}\right) \quad \ldots$$

(5) $\frac{\partial J(\theta)}{\partial a_1^{(3)}}$; (1) Binary cross Entropy ? ✓

(2) Squared Error Function ?

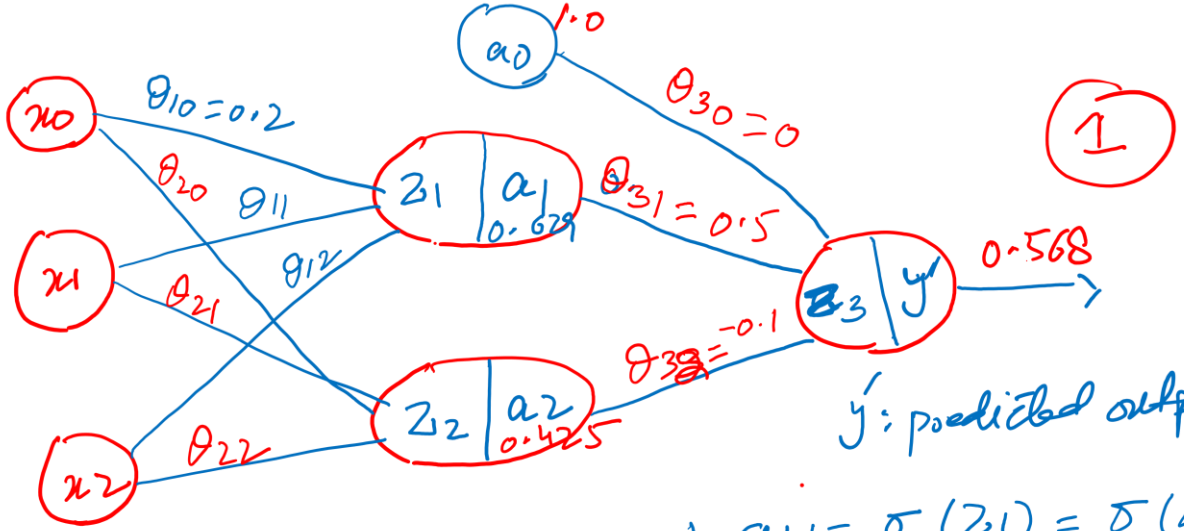
(i) $\frac{\partial J(\theta)}{\partial a_1^{(3)}} = \frac{\partial}{\partial a_1^{(3)}}\left[-y\left(\log\left(a_1^{(3)}\right)\right)-(1-y)\log\left(1-a_1^{(3)}\right)\right]$

$= -y\frac{\partial}{\partial a_1^{(3)}}\left(\log\left(a_1^{(3)}\right)\right)-(1-y)\frac{\partial}{\partial a_1^{(3)}}\left(\log\left(1-a_1^{(3)}\right)\right)$

$= \frac{-y}{a_1^{(3)}} + \frac{(1-y)}{1-a_1^{(3)}}$;

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(1)}} = \left[\frac{-y}{a_1^{(3)}} + \frac{(1-y)}{1-a_1^{(3)}}\right] \times a^{(3)}(1-a^{(3)}) \times \theta_{11}^{(2)} \times a_1^{(2)}(1-a_1^{(2)}) \times x_2$$

Neural Network Example :



$\hat{y}$: predicted output.

$z_1 = \theta_{10} \times 1 + \theta_{11} \times x_1 + \theta_{12} \times x_2$

$z_1 = 0.2 \times 1 + 0.4 \times 0.6 + 0.1 \times 0.9$

$z_1 = 0.2 + 0.24 + 0.09 = 0.53$

$a_1 = \sigma(z_1) = \sigma(0.53) = 0.629$

$z_2 = \theta_{20} \cdot 1 + \theta_{21} \times x_1 + \theta_{22} \times x_2$

$= -0.3 + 0.18 + 0.18$

$= -0.3$

$a_2 = \sigma(-0.3) \approx 0.425$

$z_3 = 0.272$

$y' = \sigma(0.272) \approx 0.568$

② $J(\theta) = E =$ Cost computation: $\frac{1}{2}(y' - t)^2$

$J(\theta) = E = \frac{1}{2} * (0.568 - 1)^2$

$J(\theta) = E = 0.093$

③ optimization / Backpropagation

$$\theta_{ji} = \theta_{ji} - \alpha \frac{\partial J(\theta)}{\partial \theta_{ji}}$$

$$\theta_{31} = \theta_{31} - \alpha \frac{\partial J(\theta)}{\partial \theta_{31}}$$

Chain Rule

$$\frac{\partial J(\theta)}{\partial \theta_{31}} = \frac{\partial J(\theta)}{\partial y'} \frac{\partial y'}{\partial z_3} \frac{\partial z_3}{\partial \theta_{31}}$$

$$\frac{\partial J(\theta)}{\partial y'} = \frac{1}{2}(y' - t)^2$$

$$= \frac{2}{2}(y' - t) \times \frac{\partial (y' - t)}{y'}$$

$$= y' - t$$

$$= 0.568 - 2$$

$$= -0.432$$

$$\frac{\partial y'}{\partial z_3} = y'(1-y')$$

$$= 0.568\,(1-0.568)$$

$$= \boxed{S_1}$$

$$\frac{\partial z_3}{\partial \theta_{31}} = \theta_{30} x_0 + \boxed{\theta_{31} \cdot a_1} + \theta_{32} a_2$$

$$= a_1 = 0.629$$

$$\frac{\partial J(\theta)}{\partial \theta_{31}} = (-0.432) \times 0.568 \times (.432) \times 0.629$$

$$= S_{31}$$

$$\frac{\partial J(\theta)}{\partial \theta_{32}}, \frac{\partial J(\theta)}{\partial \theta_{30}}, \theta_{10}, \theta_{11}, \theta_{12}, \theta_{20}, \theta_{21}, \theta_{22}$$

$\alpha = 0.01$

$\theta_{31} = \theta_{31} - \alpha (S_{31})$

$\theta_{31} = 0.5 - (0.01) \times S_{31}$

θ_{30}
θ_{32}
θ_{11}
θ_{22}

2nd iteration of
- forward propagation
- Cost.
- Backward Prop.