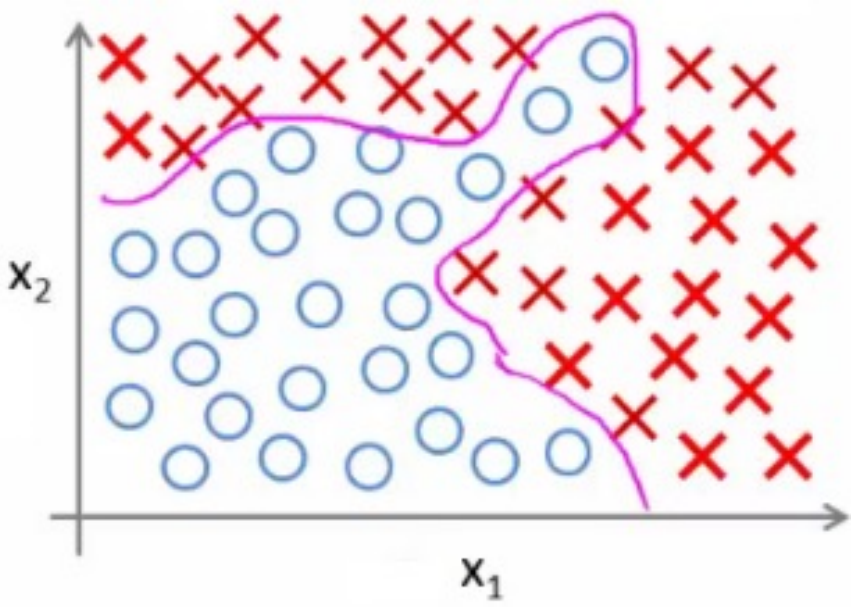
Politecnico di Bari

Neural Networks

Non-linear hypotheses

before this, we have always seen linear problem, what happen when it isnt linera? --> neural network

Non-linear Classification



a trick is using polinomial regression

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \theta_7 x_1 x_2^3 + \ldots)$$

x_1 = size
x_2= #bedrooms
x_3 = #floors
x_4 = age
…
x_{100}

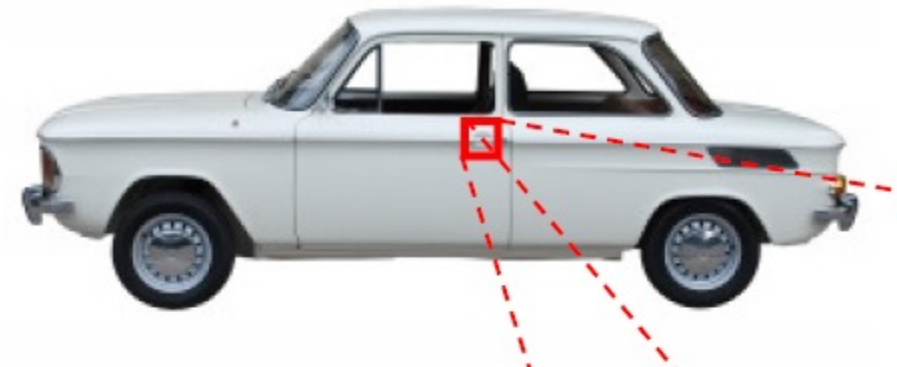
n=100

$$x_1^2, x_2^2, x_3^2, \ldots$$
$$x_1 x_2, x_1 x_3, x_1 x_4, \ldots, x_1 x_{100}$$
$$\approx \quad 5000 features \quad \mathcal{O}(n^2)$$

$$x_1^3, x_2^3, x_3^3, \ldots$$
$$x_1^2, x_2^2, x_3^2, \ldots$$
$$x_1^2 x_2, x_1^2 x_3, x_1^2 x_4, \ldots, x_1^2 x_{100}$$
$$x_1 x_2, x_1 x_3, x_1 x_4, \ldots, x_1 x_{100}$$
$$\approx \quad 170000 features \quad \mathcal{O}(n^2)$$

it grows to much, is not usable

How many features do we usually deal with?

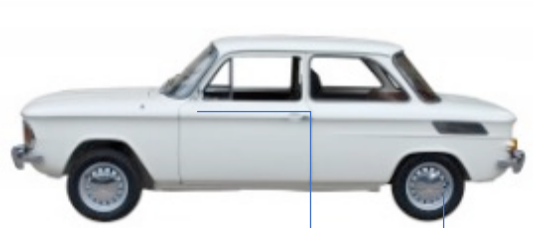


the only things the algorithm can see
is the value of the pixel. we have the same feature
as the number of pixel

But the camera sees this:

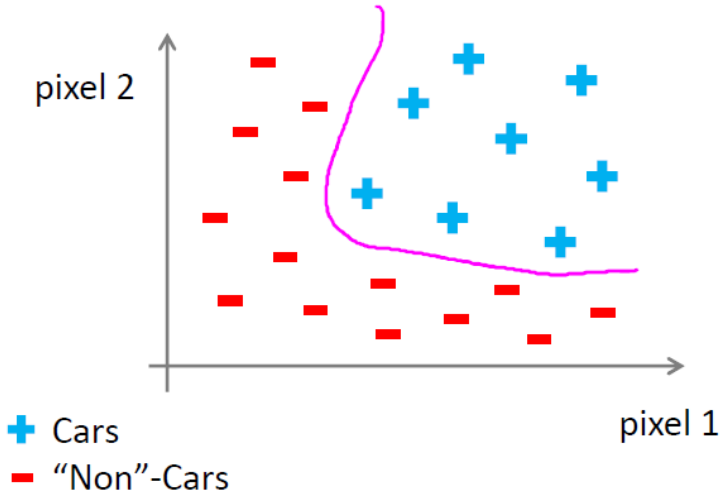
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	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

How many features do we usually deal with? (cont.d)



small image

Simple 50 x 50 pixels images imply 2500 features

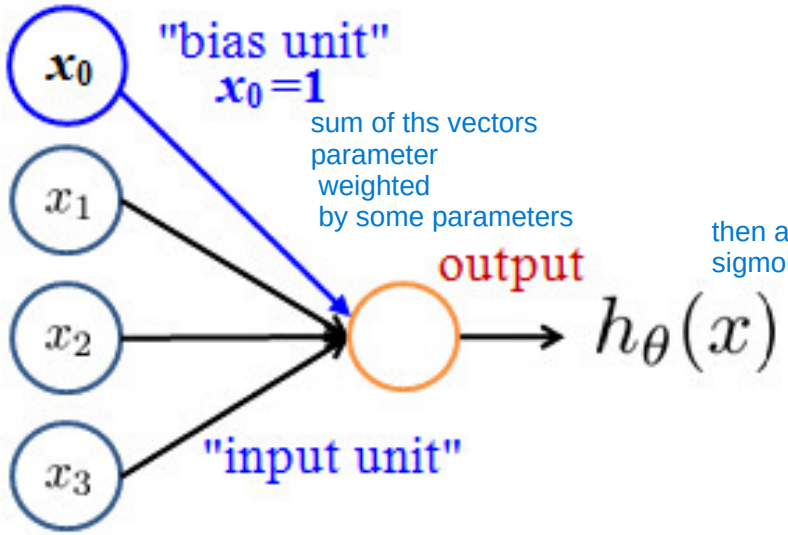


$$\mathbf{x} = \begin{bmatrix} \text{pixel} & 1 & \text{intensity} \\ \text{pixel} & 2 & \text{intensity} \\ & \vdots & \\ \text{pixel} & 2500 & \text{intensity} \end{bmatrix}$$

If we consider the quadratic features we get $O(2.500^2) = O(6 \cdot 10^6)$ features

if we use quadratic regression. we would have 6 million feature

Neuron model



sum of ths vectors parameter weighted by some parameters

then apply the sigmoid function

vector of input

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

weights

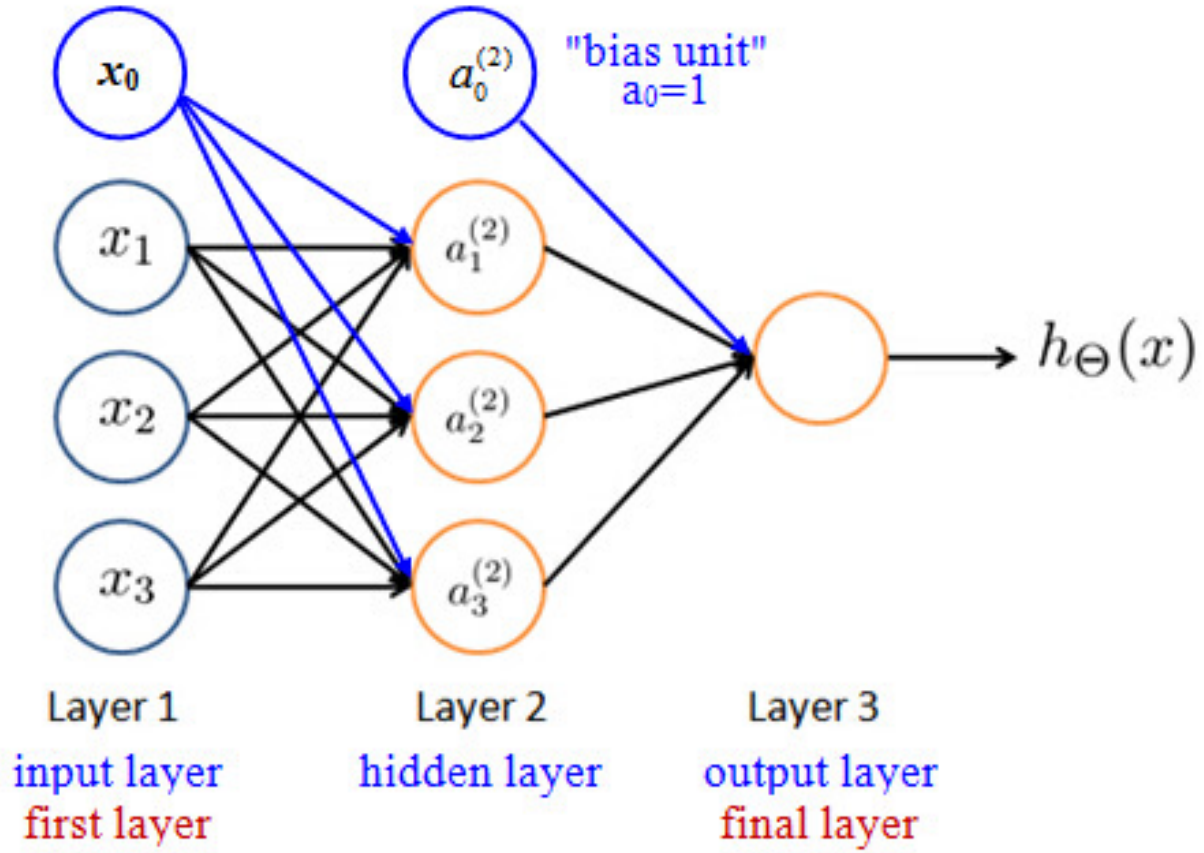
Sigmoid (logistic) activation function.

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

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

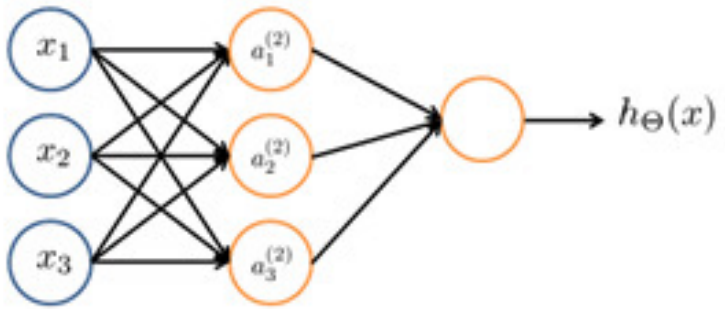
Neural Network

we can combine several neuron creating a network



between input and output layer we can have several hidden layer

Neural Network (cont.d)



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

the output of a_1^(2) is the logistic regression applied to the linear combination of the inputs plus the bias

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

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

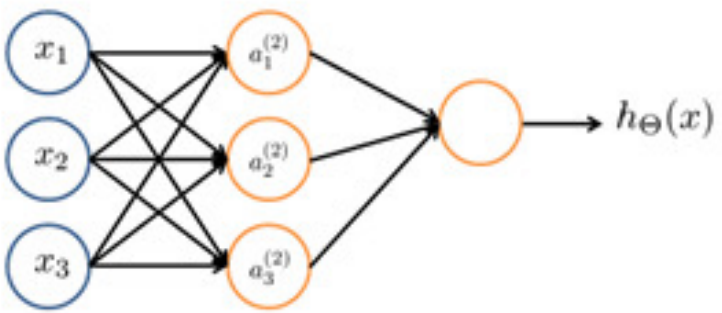
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

the output

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

in general the output of each node layer is the logistic regression applied tot the linear combination of the inputs of the node of the previous layer plus the bias

Neural Network (cont.d)



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

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

we have a matrix of parameters

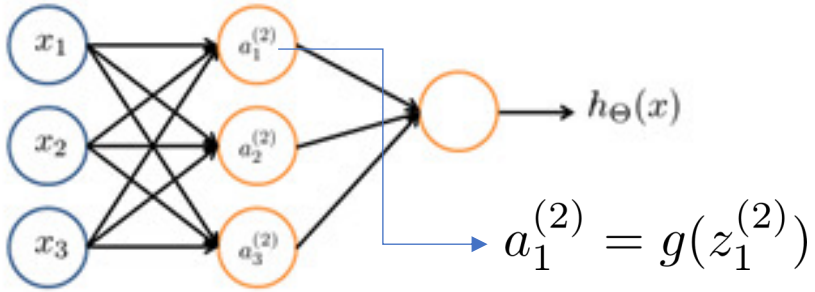
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2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

$$\Theta^{(1)} = \begin{bmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} & \Theta_{13}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} & \Theta_{23}^{(1)} \\ \Theta_{30}^{(1)} & \Theta_{31}^{(1)} & \Theta_{32}^{(1)} & \Theta_{33}^{(1)} \end{bmatrix}$$

$$\Theta^{(2)} = \begin{bmatrix} \Theta_{10}^{(2)} & \Theta_{11}^{(2)} & \Theta_{12}^{(2)} & \Theta_{13}^{(2)} \end{bmatrix}$$

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

Forward propagation: Vectorized implementation



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

$z_1^{(2)}$

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

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

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

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

$$\Theta^{(1)} = \begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{bmatrix}_{3\times 4} \quad \Theta^{(2)} = \begin{bmatrix} \theta_{10}^{(2)} & \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \end{bmatrix}_{1\times 4}$$

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

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

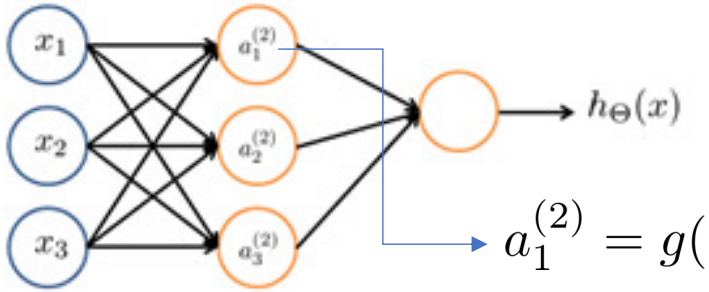
$$a^{(2)} = g(z^{(2)})$$

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

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

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

Forward propagation: Vectorized implementation



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

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$
$$h_\Theta(x) = a_1^{(3)} = g(z_1^{(3)})$$

$$\Theta^{(1)} = \begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{bmatrix}_{3\times 4} \quad \Theta^{(2)} = \begin{bmatrix} \theta_{10}^{(2)} & \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \end{bmatrix}_{1\times 4}$$

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{z^{(2)}} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \quad \mathbf{a^{(2)}} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix}$$

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

Politecnico di Bari

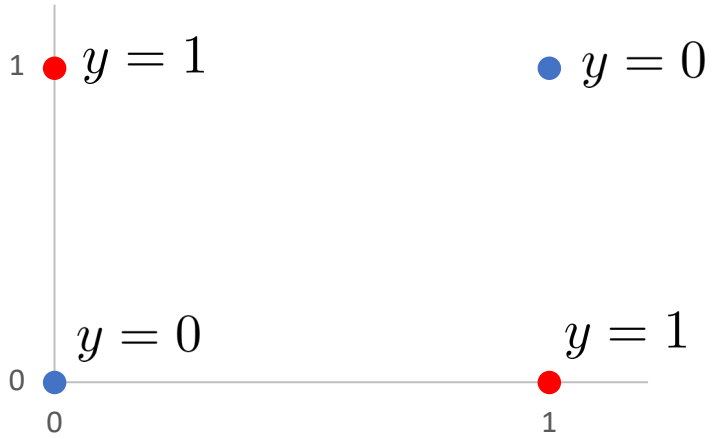
Neural Networks

Intuition

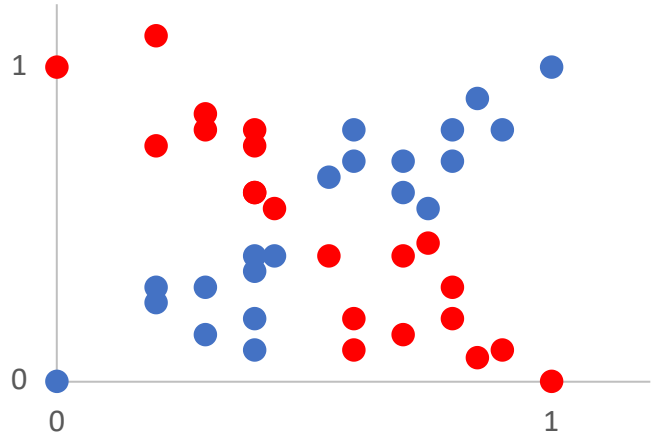
Non-linear classification example:XOR

Linear classifers are not able to solve it

X1 and x2 are binary (0 or 1)



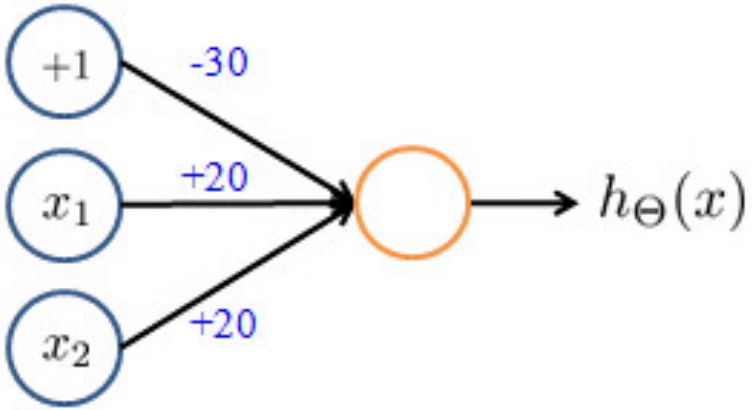
y= x1 XOR x2



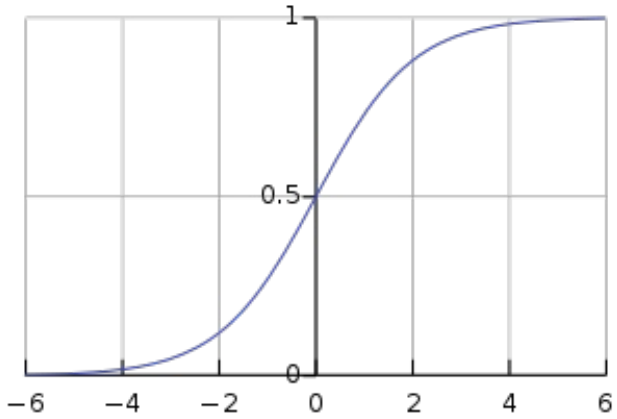
NN example: AND function

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

$y = x_1 \text{AND} x_2$



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

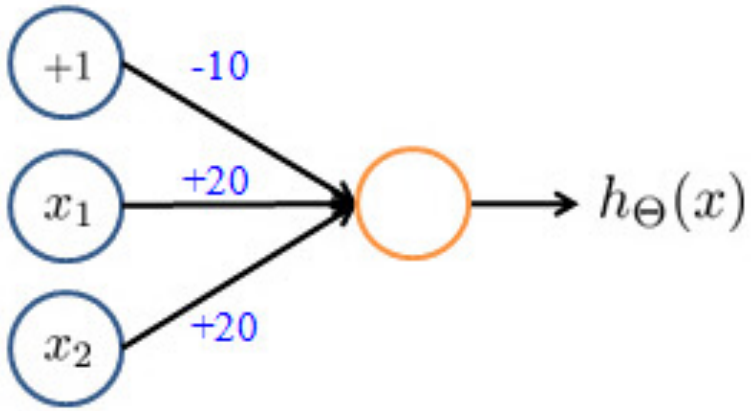


x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(+10) \approx 1$

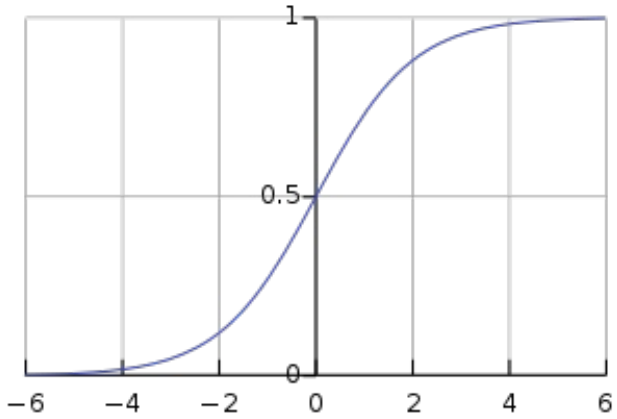
NN example: OR function

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

$y = x_1 \mathrm{OR} x_2$



$h_\Theta(x) = g(-10 + 20x_1 + 20x_2)$

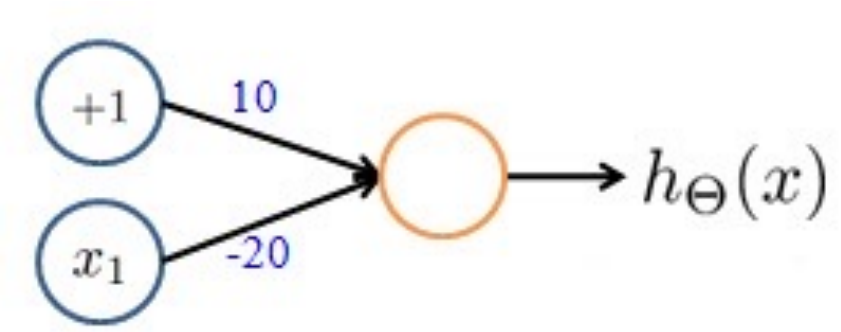


x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(+10) \approx 1$
1	0	$g(+10) \approx 1$
1	1	$g(+30) \approx 1$

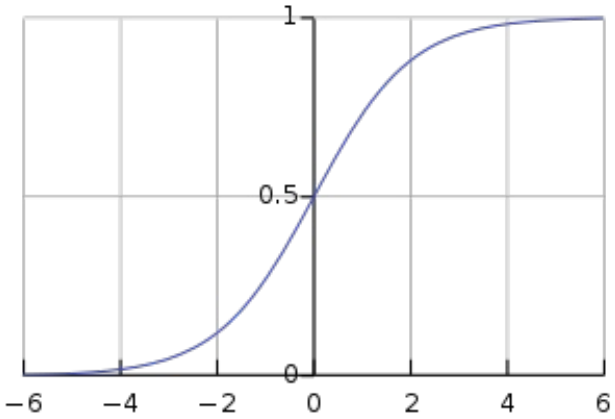
NN example: NOT function

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

$y = \text{NOT } x_1$



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

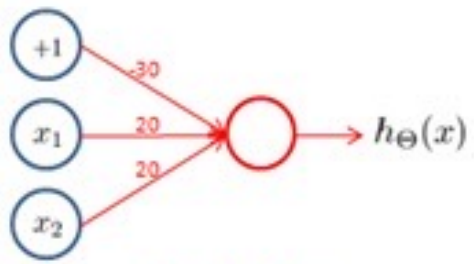


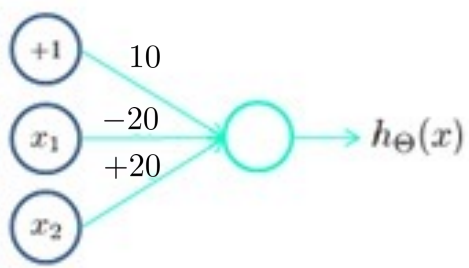
x_1	$h_\Theta(x)$
0	$g(+10) \approx 1$
1	$g(-10) \approx 0$

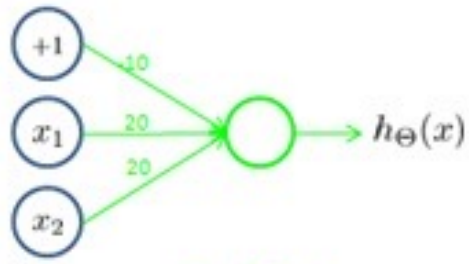
NN example: XOR function

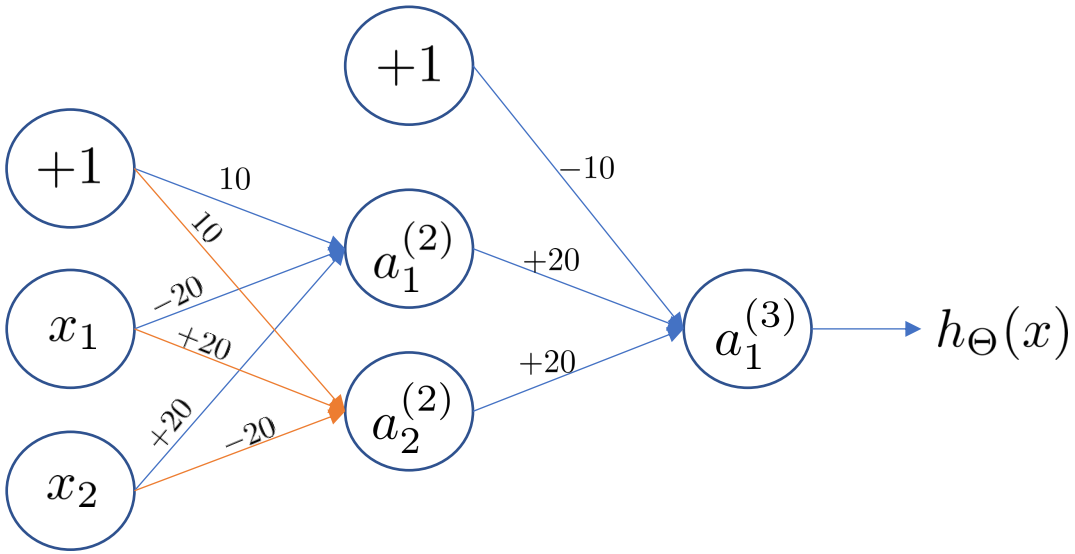
$x_1 \in \{0, 1\}$

$y = x_1 \text{XOR } x_2$









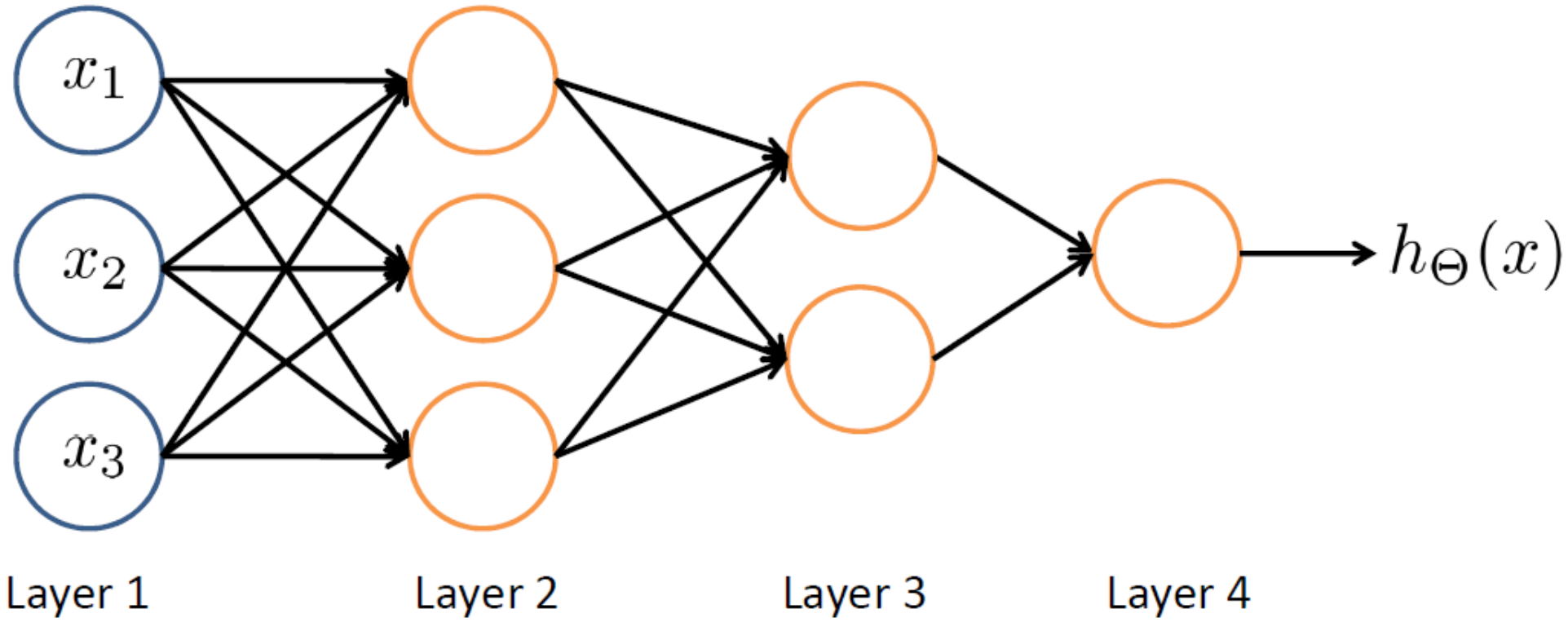
just 1 hidden layer

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

$x_1 \text{XOR } x_2 = \big(x_1\ AND\ (NOT\ x_2)\big) OR\ (x_2\ AND\ NOT\ x_1)$

Neural Networks

this is the example we are going to use, if i have only one output node it meas that is a binary classification



Neural Networks for multi-class classification

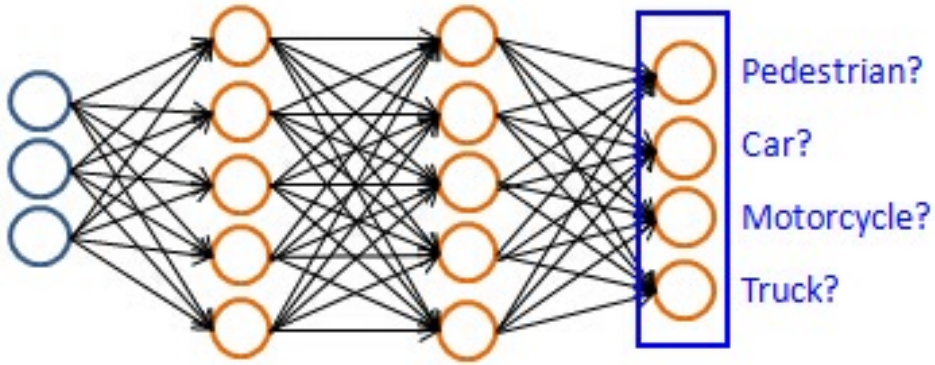

Pedestrian


Car


Motorcycle


Truck

in this case we have 4 class to predict, so we have 4 output node



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

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 pedestrian when car when motorcycle

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

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

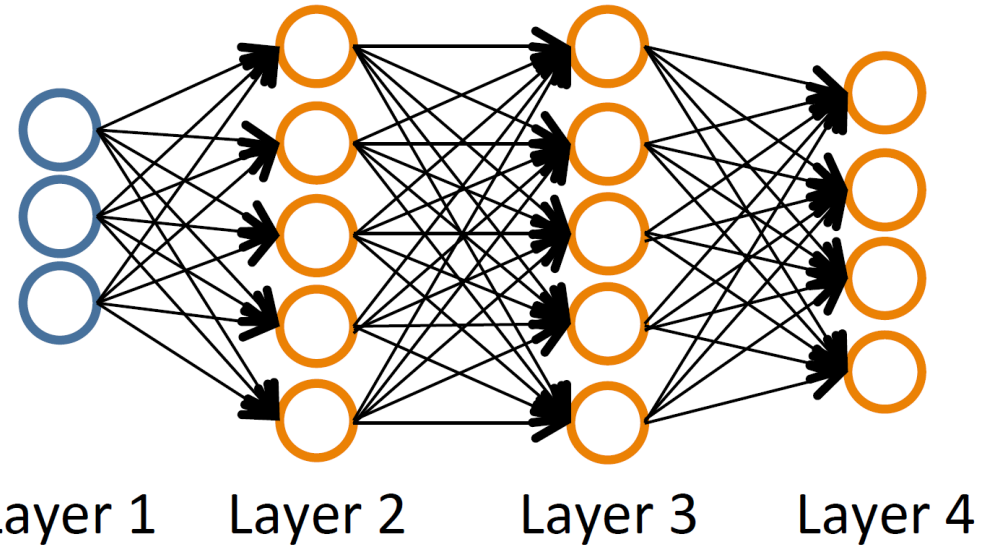
pedestrian car motorcycle truck

Politecnico di Bari

Neural Networks

Cost function

Neural Network for classification task



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

L = total number of layers in network

s_l = number of units (but the bias unit) in layer l

Binary classification

$$y = 0 \text{ or } 1$$

1 output unit

Multi-class classification (k classes)

$$y \in \mathbb{R}^K = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

K output units

Cost function for Classification

Logistic regression

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

Neural Network

$$h_\Theta(x) \in \mathbb{R}^K \qquad \text{with } \left(h_\Theta(x)\right)_k = k^{th} \text{ output}$$

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

sum of all the sample

sum of all classes

sum of all the layer

sum of all the column of the matrix

sum of all the row of the theta matrix

L2 norm,

Cost function for Regression

Linear regression

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

Neural Network

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\theta_{ji}^2$$

compute the error on the output layer

Politecnico di Bari

Neural Networks

Backpropagation algorithm

Gradient computation for Classification

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

To minimize $J(\Theta)$: $\min_{\Theta} J(\Theta)$

We need to compute:

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

Gradient computation for Regression

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\theta_{ji}^2$$

To minimize $J(\Theta)$: $\min_{\Theta} J(\Theta)$

We need to compute:

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

Gradient computation

Given the training example (x,y):

Forward propagation:

$a^{(1)} = x$ input

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

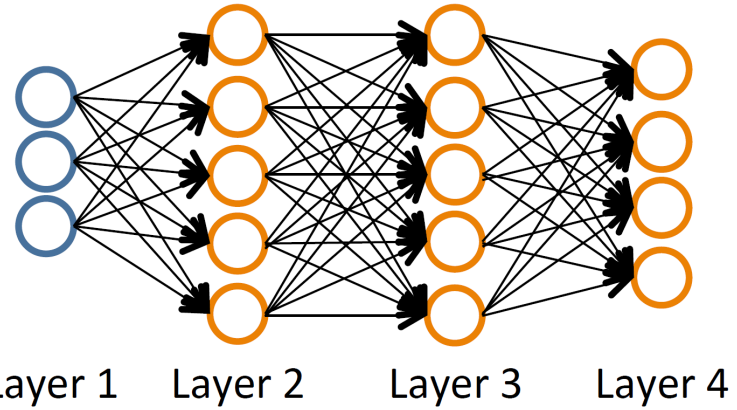
$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$

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

$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$

$z^{(4)} = \Theta^{(3)} a^{(3)}$

$a^{(4)} = h_\Theta(x) = g(z^{(4)})$



Gradient computation – Regression Loss

without regolarization

$$\frac{\partial J}{\partial \Theta^{(3)}} = \frac{\partial \left(\frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})^2 \right)}{\partial \Theta^{(3)}}$$

the last matrix

$$= (a^{(4)} - y) \frac{\partial g(z^{(4)})}{\partial \Theta^{(3)}}$$

$$= (a^{(4)} - y) g'(z^{(4)}) \frac{\partial z^{(4)}}{\partial \Theta^{(3)}} \text{ with } z^{(4)} = \Theta^{(3)} a^{(3)}$$

the partial derivative of z4 is a3

$$= \delta^{(4)} a^{(3)}$$

encoded the error, pushing back the error

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

Gradient computation – Regression Loss

$$
\begin{aligned}
\frac{\partial J}{\partial \Theta^{(2)}} &= \delta^{(4)} \frac{\partial z^{(4)}}{\partial \Theta^{(2)}} \\
&= \delta^{(4)} \frac{\partial \Theta^{(3)} a^{(3)}}{\partial \Theta^{(2)}} \\
&= \Theta^{(3)} \delta^{(4)} \frac{\partial a^{(3)}}{\partial \Theta^{(2)}} \\
&= \Theta^{(3)} \delta^{(4)} \frac{\partial g(z^{(3)})}{\partial \Theta^{(2)}} \text{ with } a^{(3)} = g(z^{(3)}) \\
&= \Theta^{(3)} \delta^{(4)} g'(z^{(3)}) \frac{\partial z^{(3)}}{\partial \Theta^{(2)}} \text{ with } z^{(3)} = \Theta^{(2)} a^{(2)} \\
&= \delta^{(3)} a^{(2)}
\end{aligned}
$$

delta 3 encoded also delta 4

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

Gradient computation – Regression Loss

$$\frac{\partial J}{\partial \Theta^{(1)}} = \delta^{(3)} \frac{\partial z^{(3)}}{\partial \Theta^{(1)}}$$

$$= \Theta^{(2)} \delta^{(3)} g'(z^{(2)}) \frac{\partial z^{(2)}}{\partial \Theta^{(1)}}$$

$$= \delta^{(2)} a^{(1)}$$

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

Gradient computation – Regression Loss

Focus on the last error contribution:

$$\delta_j^{(4)} \propto a_j^{(4)} - y_j \longrightarrow h_\Theta(x)$$

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

In this example lambda will be considered 0

Get an idea of the gradient:

$$\frac{\partial}{\partial \theta_k} J(\theta) = \left(h_\theta(\mathbf{x}^{(\mathbf{i})}) - y^{(i)} \right) x_k^{(i)} \longrightarrow \text{Logistic regression gradient}$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

Gradient computation – Regression Loss

Intuition: $\delta_j^{(l)}$ = «error» of node j in layer l.

For each output unit (layer L = 4)

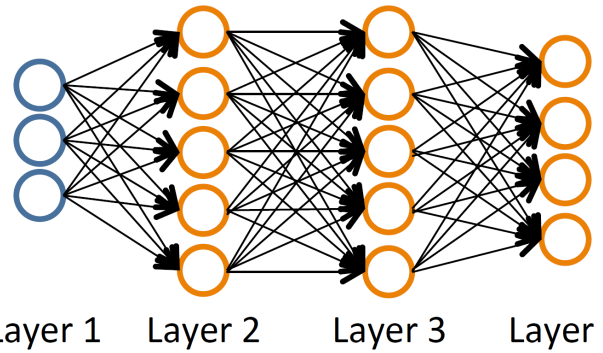
$$\delta_j^{(4)} = a_j^{(4)} - y_j \qquad \delta^{(4)} = a^{(4)} - y$$

$h_\Theta(x)$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)}) \quad \rightarrow \quad a^{(3)} .* (1 - a^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)}) \quad \rightarrow \quad a^{(2)} .* (1 - a^{(2)})$$

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



Backpropagation algorithm

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

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

For *i=1* to *m*

- Set $a^{(1)} = x^{(1)}$
- Perform forward propagation to compute $a^{(l)}$ for $l = 2,3,\ldots,L$
- Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$
- $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$$D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} \quad \text{if } j = 0$$

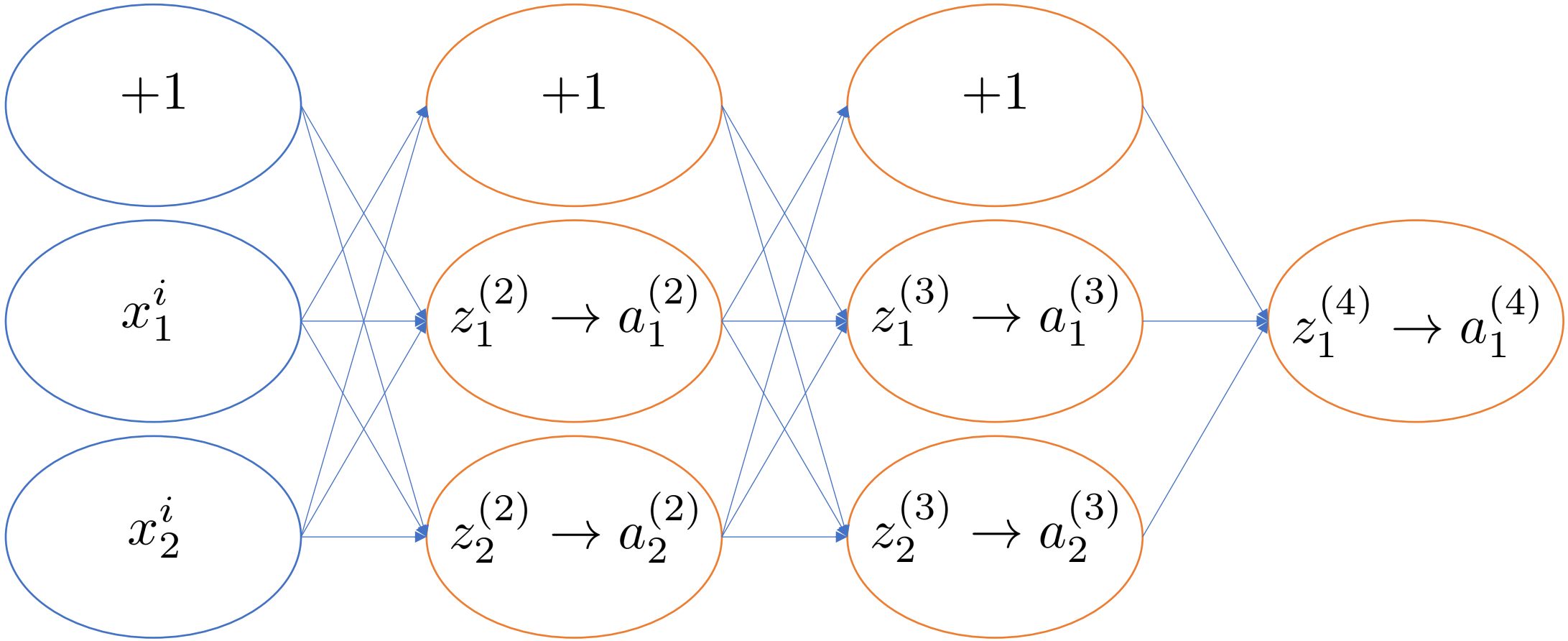
$$\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

Politecnico di Bari

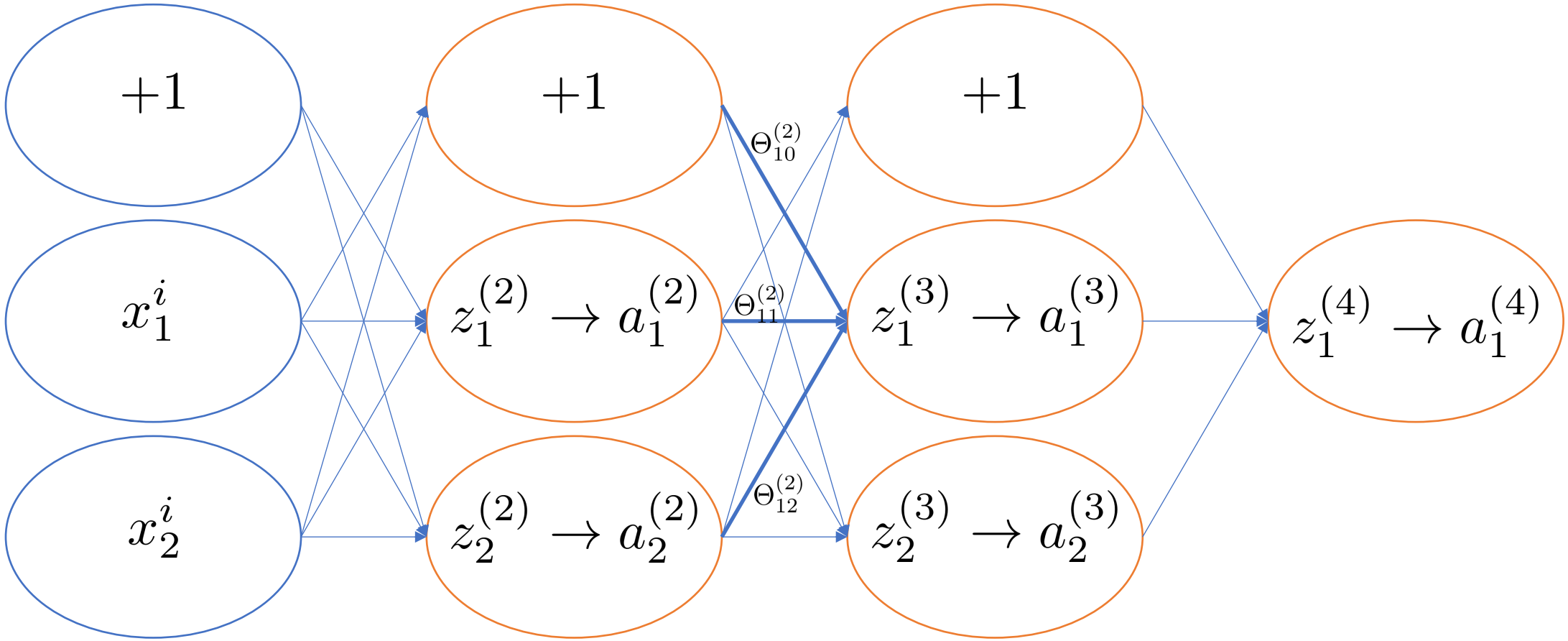
Neural Networks

Backpropagation intuition

Forward Propagation



Forward Propagation

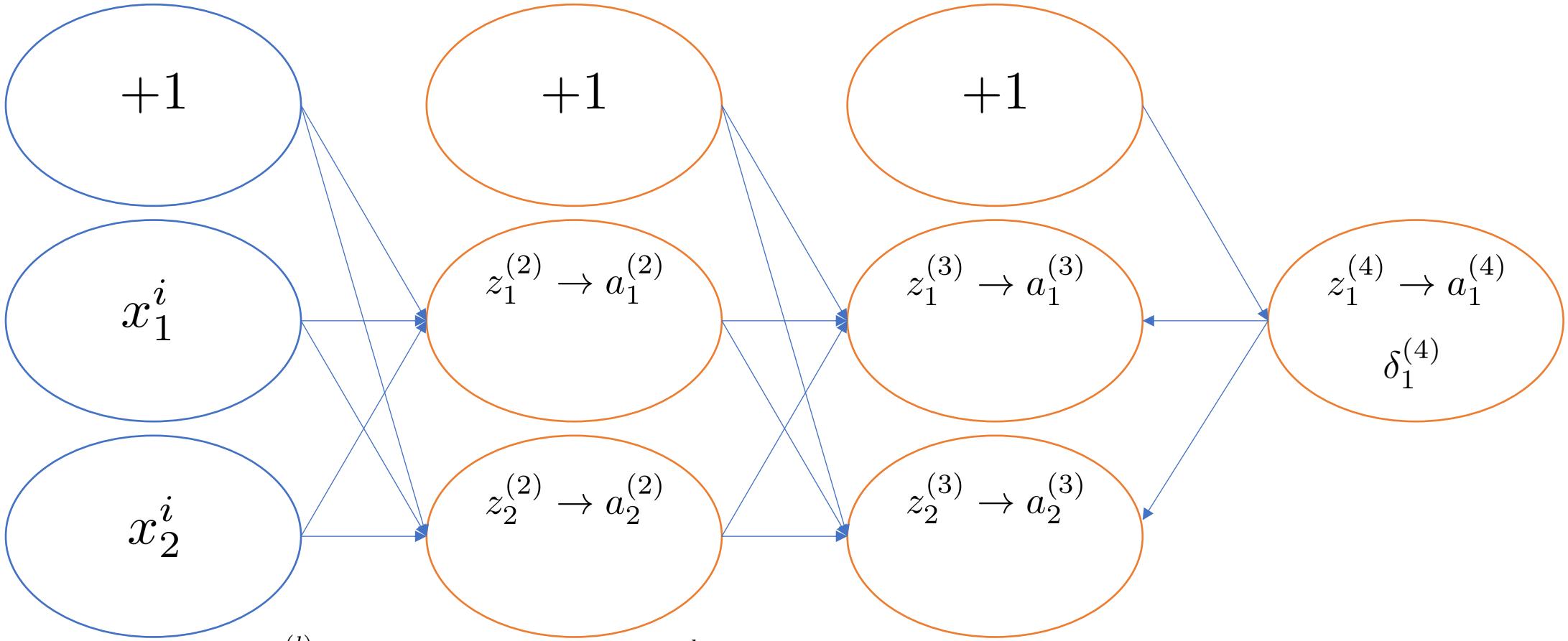


$(x^{(i)}, y^{(i)})$

$$z_1^{(3)} = \Theta_{10}^{(2)} \times 1 + \Theta_{11}^{(2)} \times a_1^{(2)} + \Theta_{12}^{(2)} \times a_2^{(2)}$$

Backward Propagation

$$\delta_1^{(4)} = y^{(i)} - a_1^{(i)}$$



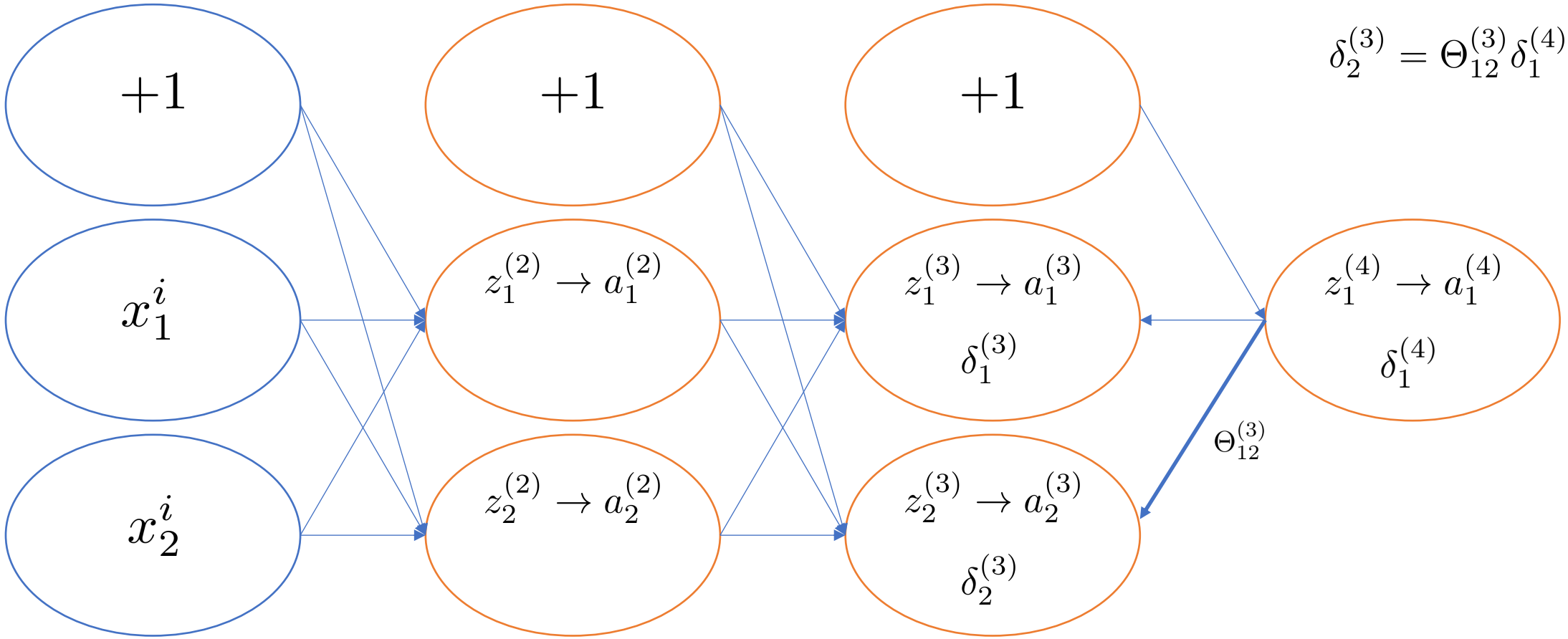
$\delta_j^{(l)} =$ error contribution for a_j^l unit j in layer l

Formally $\delta_j^{(l)} = \dfrac{\partial}{\partial z_j^{(l)}} cost(i)$ for $j \geq 0$ with $cost(i) = y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)$

Backward Propagation

$$\delta_1^{(4)} = y^{(i)} - a_1^{(i)}$$

$$\delta_2^{(3)} = \Theta_{12}^{(3)} \delta_1^{(4)}$$

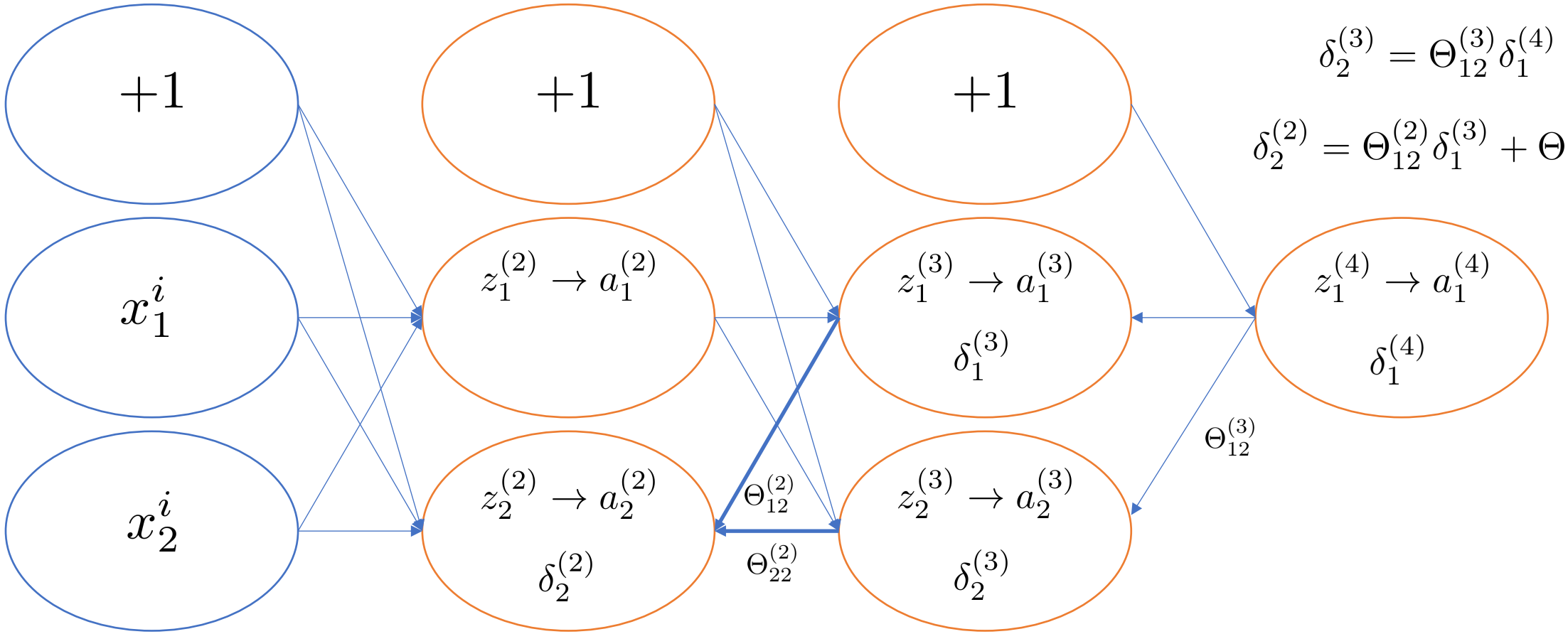


Backward Propagation

$$\delta_1^{(4)} = y^{(i)} - a_1^{(i)}$$

$$\delta_2^{(3)} = \Theta_{12}^{(3)} \delta_1^{(4)}$$

$$\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{22}^{(2)} \delta_2^{(3)}$$

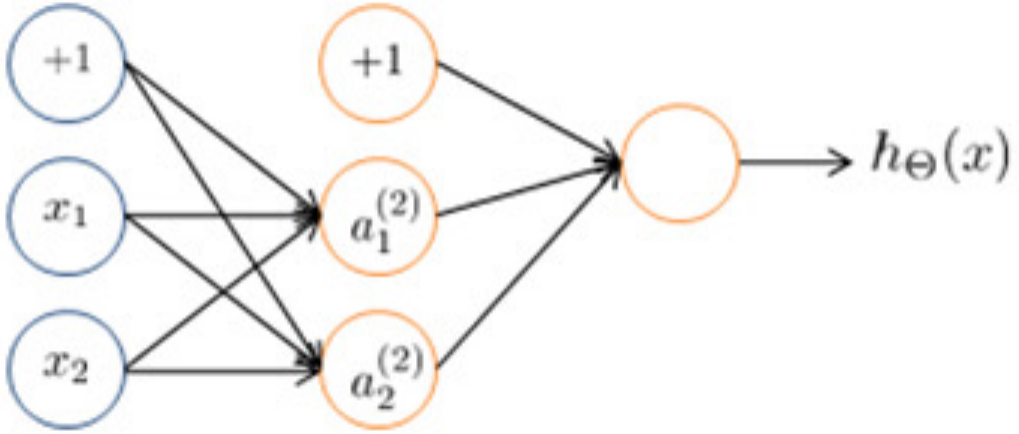


Politecnico di Bari

Neural Networks

Random initialization and Activation Functions

Zero initialization



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

Can we set the initial values of Theta to zero?

What happens?

$$a_1^2 = a_2^2$$
$$\delta_1^2 = \delta_2^2$$
$$\frac{\partial}{\partial \Theta_{01}^1} J\Theta = \frac{\partial}{\partial \Theta_{02}^1} J\Theta$$
$$\Theta_{01}^1 = \Theta_{02}^1$$

After each update

all parameters are identical !!

Random initialization

In order to break the symmetry problem

One simple solution can be to initialize the theta parameters to really small random values:

$$\Theta_{ij}^{(l)} \in [-\epsilon, \epsilon]$$

Random initialization (a bit more…)

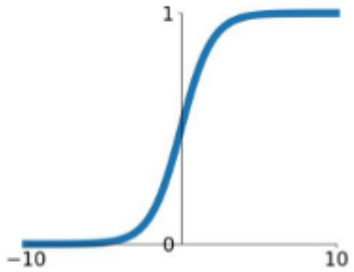
- To prevent the gradients of the network's activations from vanishing or exploding, we will stick to the following rules of thumb:
  - The mean of the activations should be zero.
  - The variance of the activations should stay the same across every layer.

- Possible solution: select values with respect to a probability distribution
  - Standard Normal
  - Uniform
  - Xavier (this is the most effective)

Activation Functions

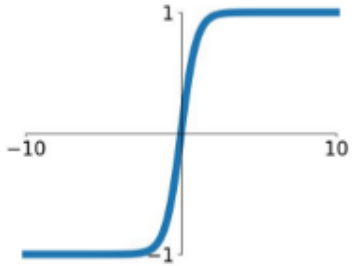
Sigmoid

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



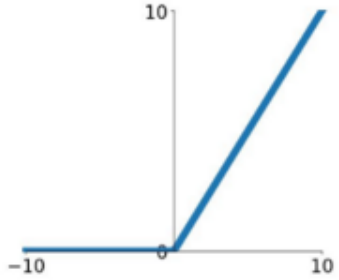
tanh

$$\tanh(x)$$



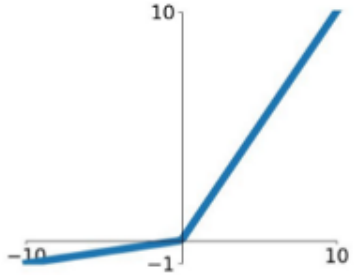
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

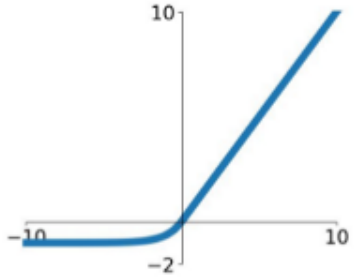


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



is always derivable

Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$