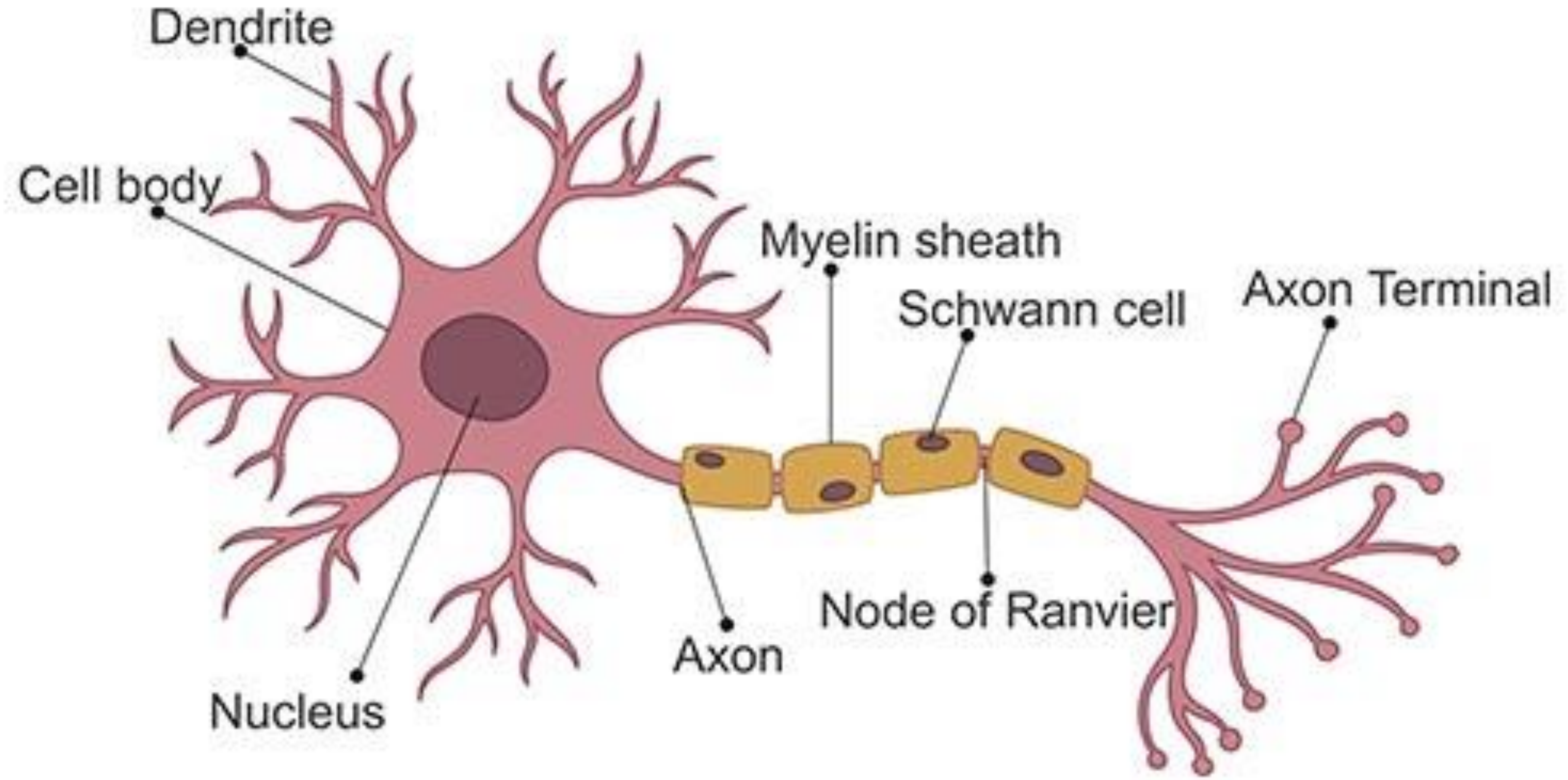
ELTE | FACULTY OF INFORMATICS

NEURAL NETWORKS

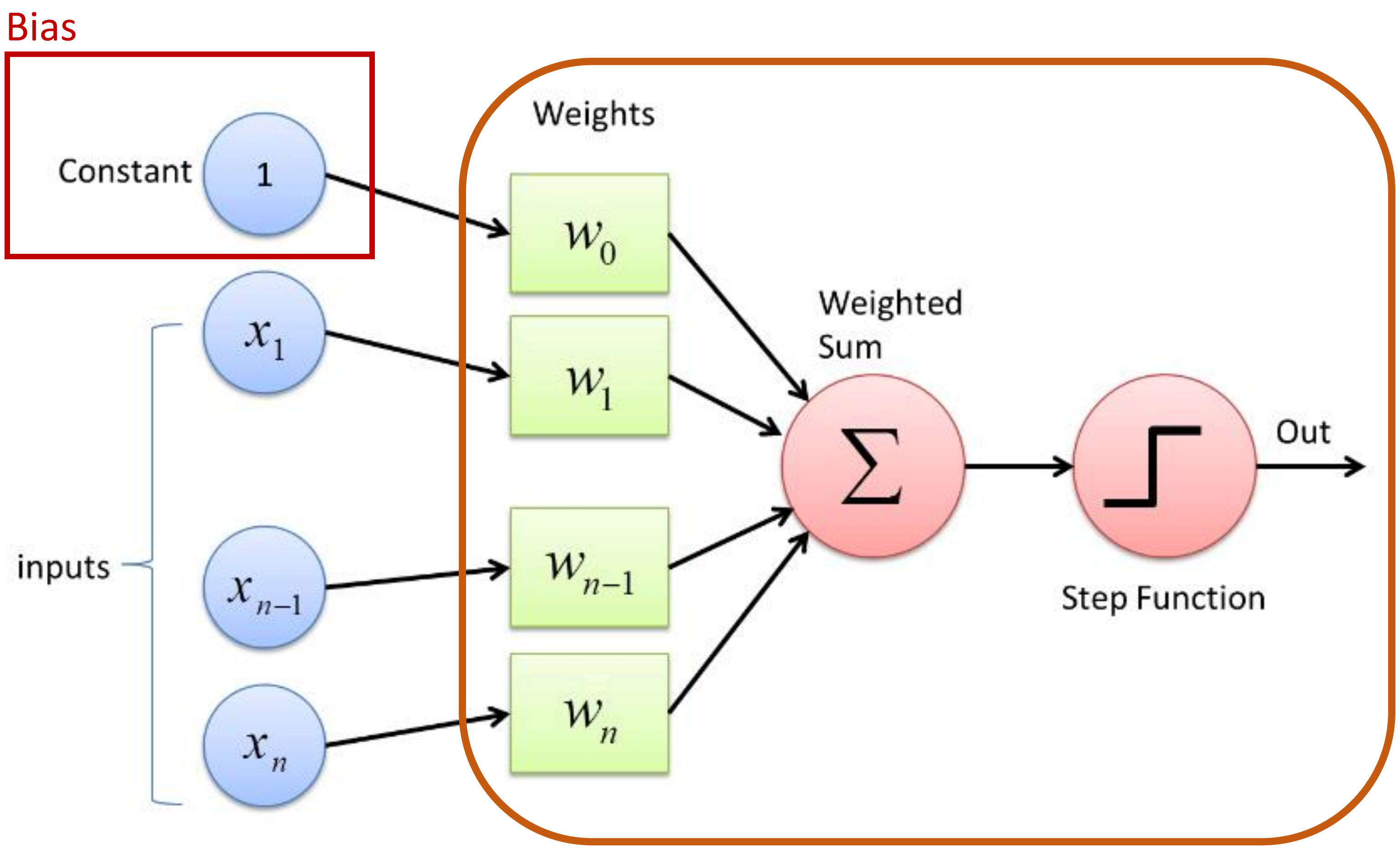
Machine Learning Course
Balázs Nagy, PhD

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

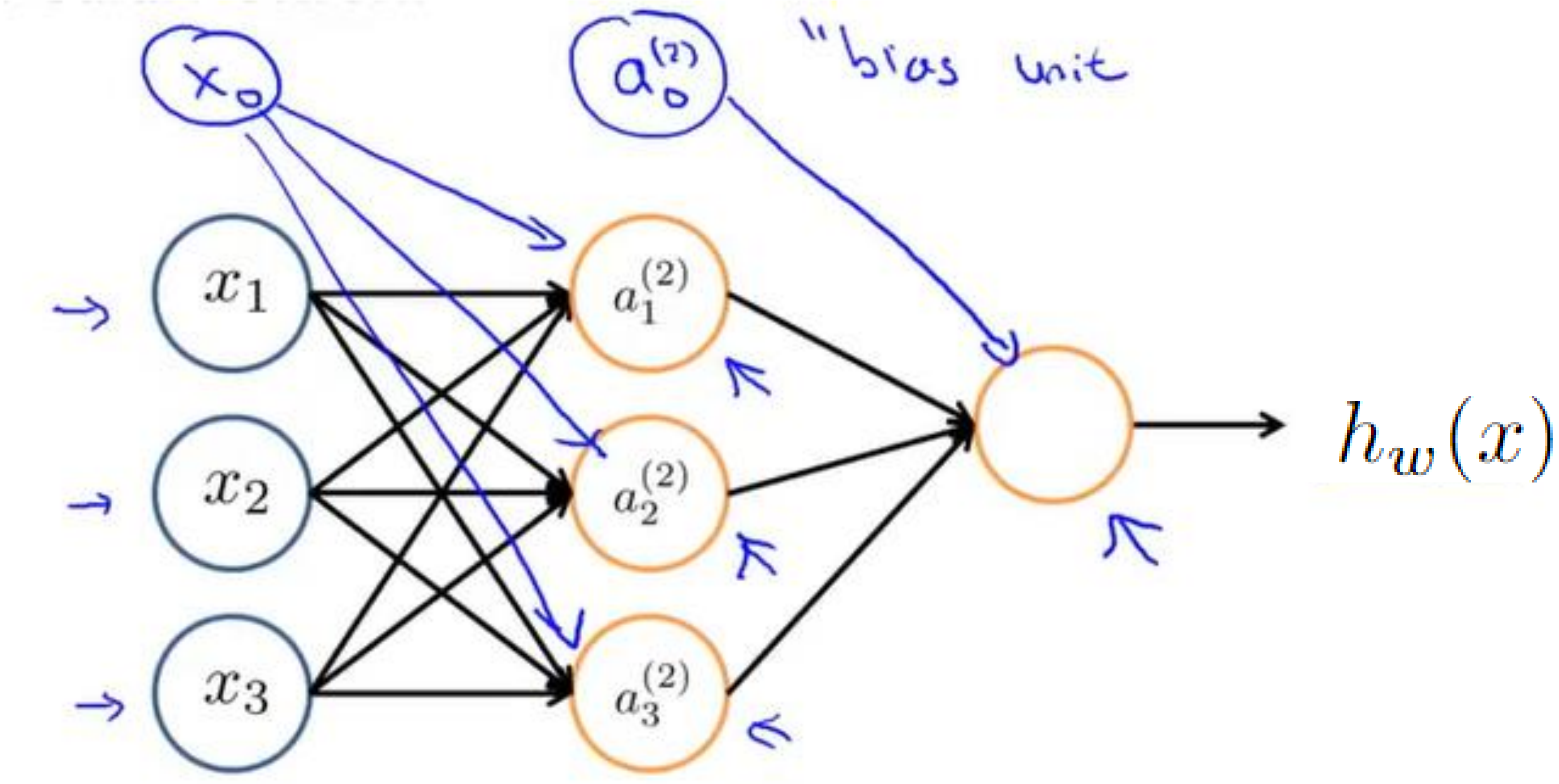
Neuron model



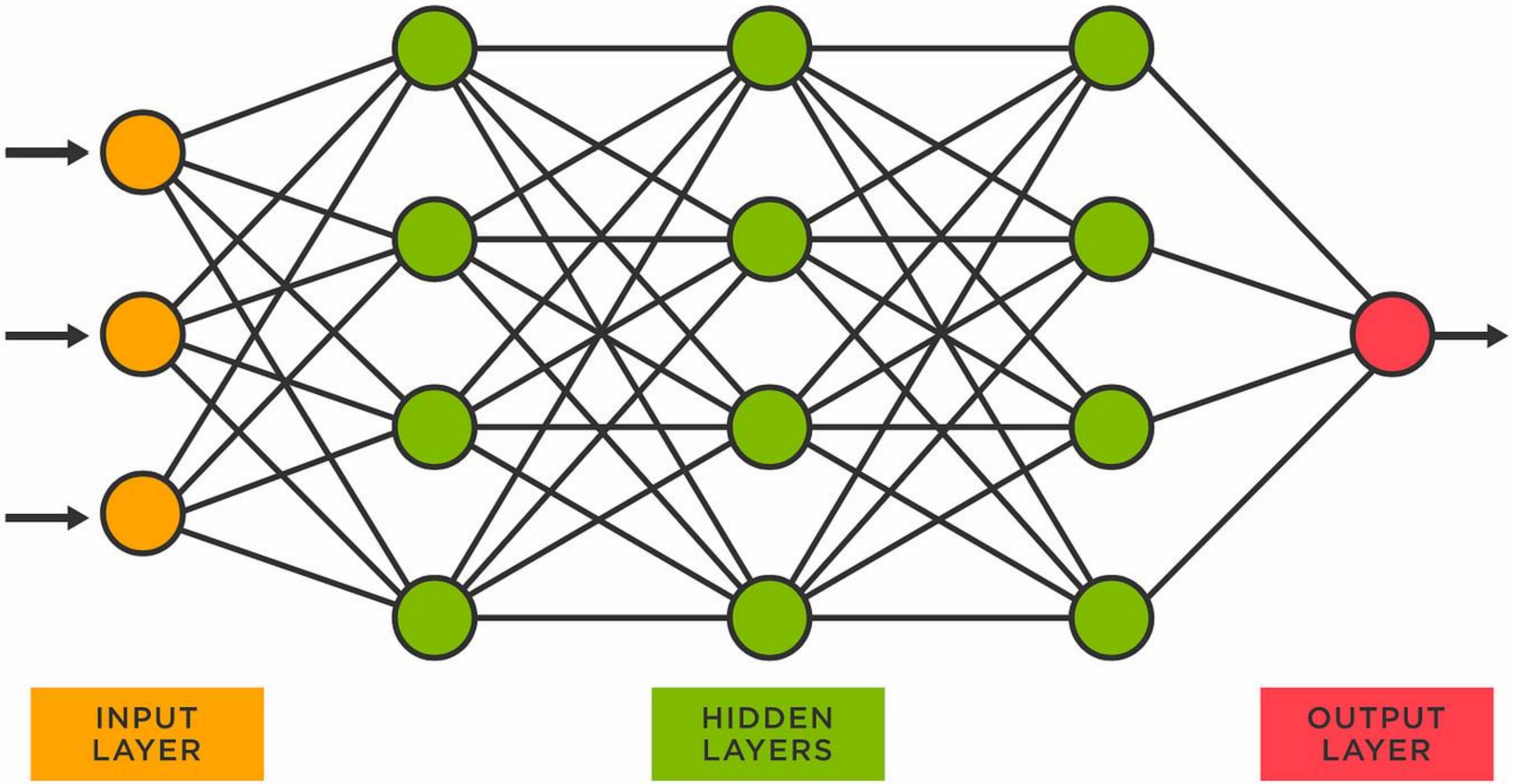
Neuron model: Perceptron



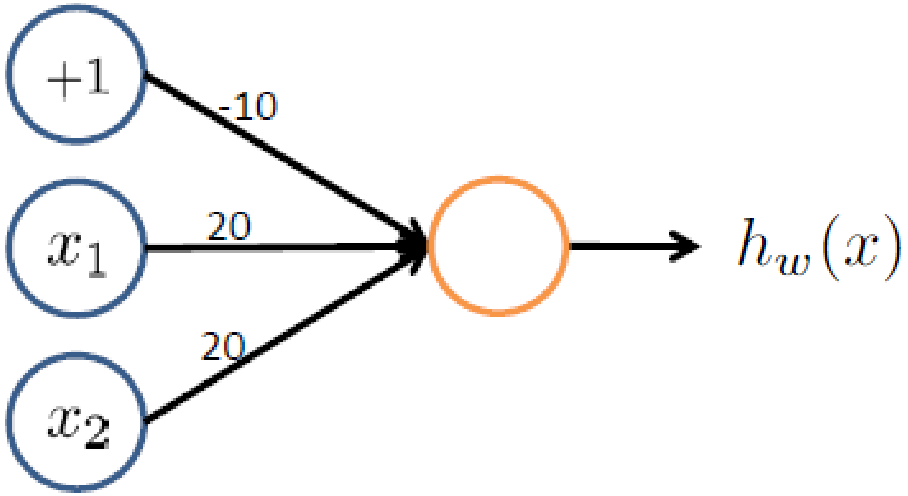
Neural Network Model



Neural Network Architecture

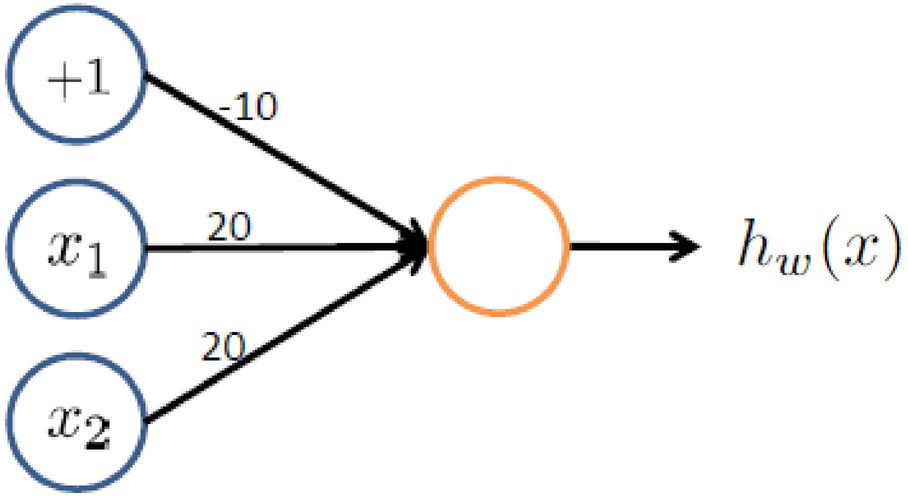


Example



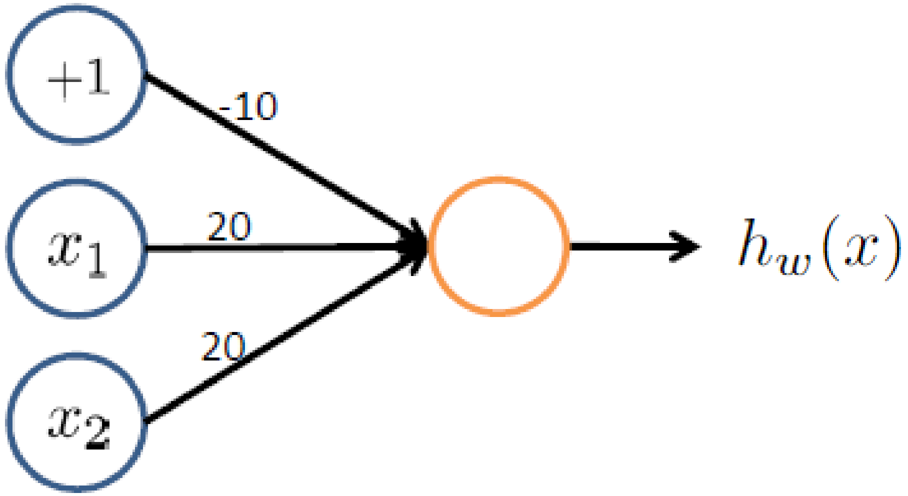
x_1	x_2	$h_w(x)$
0	0	
0	1	
1	0	
1	1	

Example



x_1	x_2	$h_w(x)$
0	0	-10
0	1	10
1	0	10
1	1	30

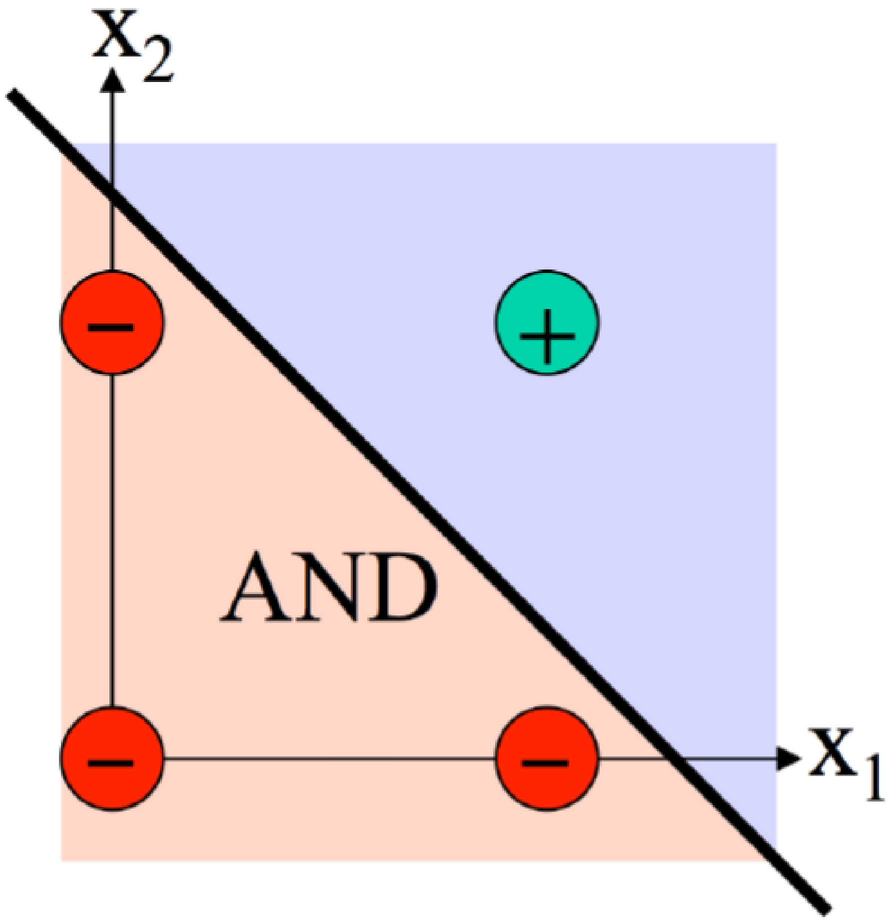
Example



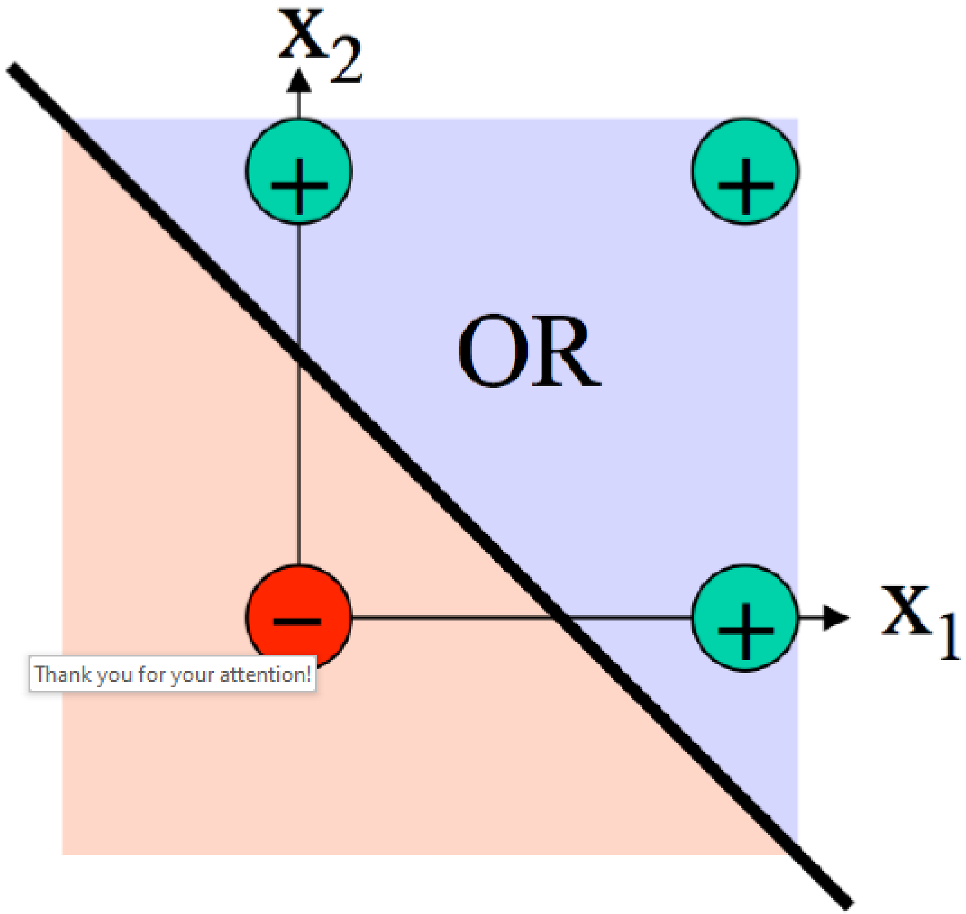
x_1	x_2	$h_w(x)$	
0	0	-10	0
0	1	10	1
1	0	10	1
1	1	30	1

If the treshold is 0
it result in the OR
logic gate

Linear separability



and.tiff



Thank you for your attention!

Linear separability

NOT

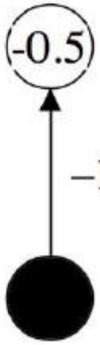
in	out
0	1
1	0

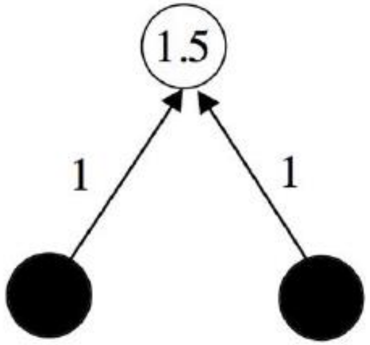
AND

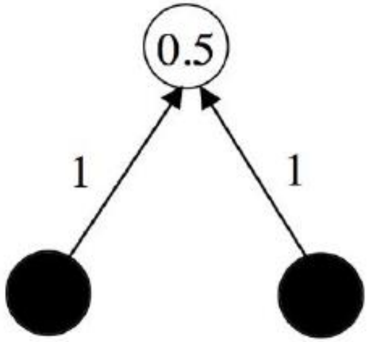
in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

OR

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1



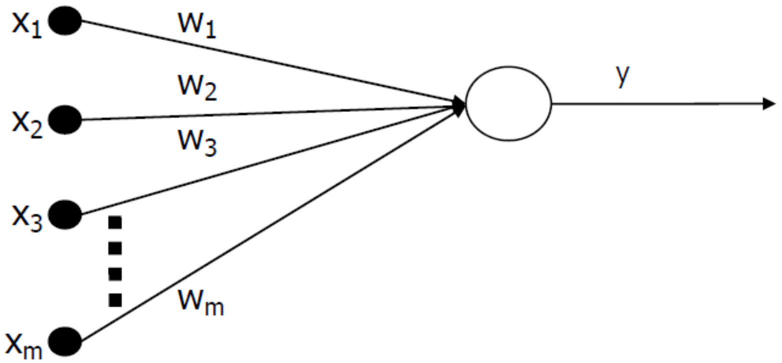




Training a Perceptron

Steps:

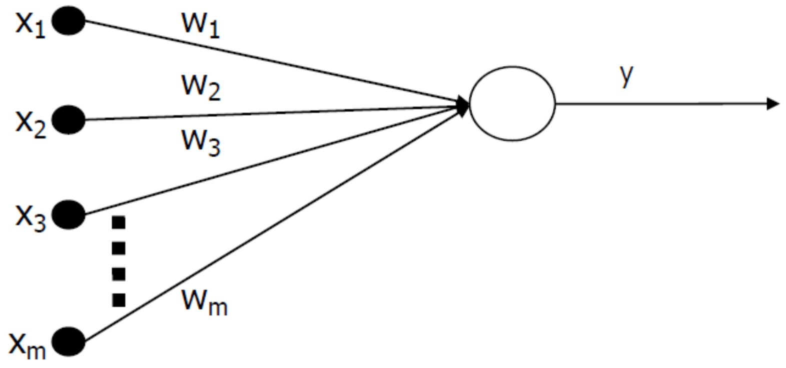
- Initialization
- Activation
- Weight update
- Iteration



Training a Perceptron

Steps:

- Initialization
- Activation
- Weight update
- Iteration

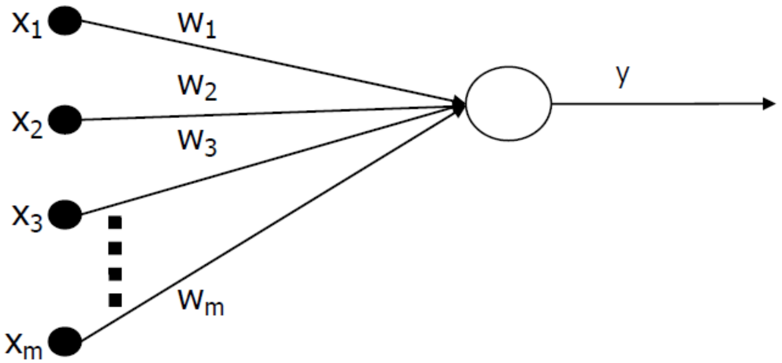


- Set the initial weights (w)
- Set the treshold value (θ) for a random variable between [-0.5, 0.5]
- Set the learning rate (η) between [0, 1]

Training a Perceptron

Steps:

- Initialization
- Activation
- Weight update
- Iteration



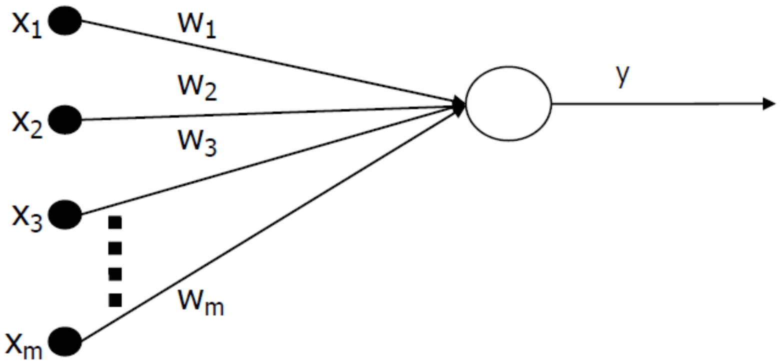
- Calculate the output in the first (p=1) iteration
- Use a predefined activation function (Φ) like step or sigmoid function

$$y = \Phi\left(\sum_{i=1}^{m} x_i \cdot w_i\right)$$

Training a Perceptron

Steps:

- Initialization
- Activation
- Weight update
- Iteration



- Update the weights

Weight update: $w_i(p+1) = w_i(p) + \Delta w_i(p)$

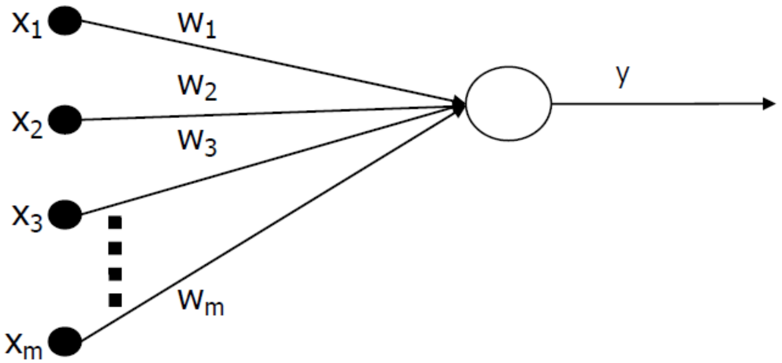
Weight correction: $\Delta w_i(p) = \eta . x_i(p) . e(p)$

Error: $e(p) = d(p) - y(p)$

Training a Perceptron

Steps:

- Initialization
- Activation
- Weight update
- Iteration



- Increment p with 1 and do the same calculations
- Do the process until convergence

p=2

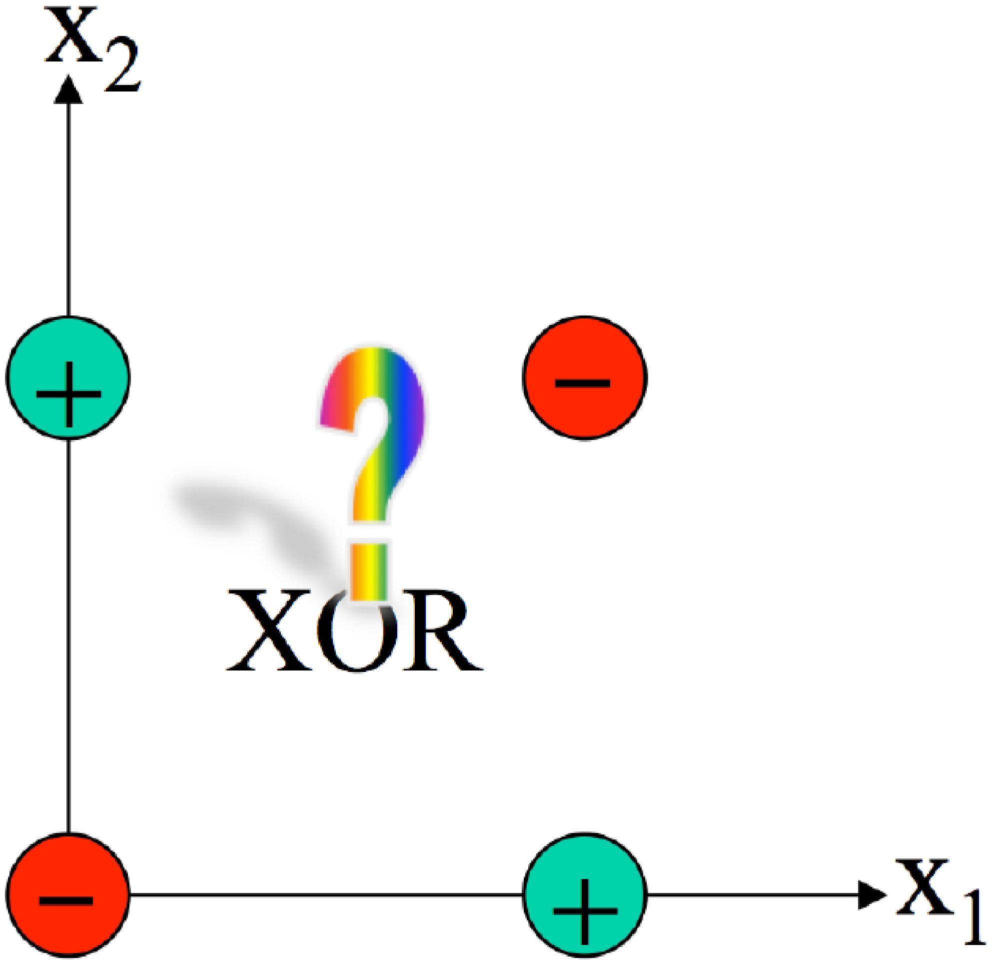
$$y = \Phi\left(\sum_{i=1}^{m} x_i \cdot w_i\right)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

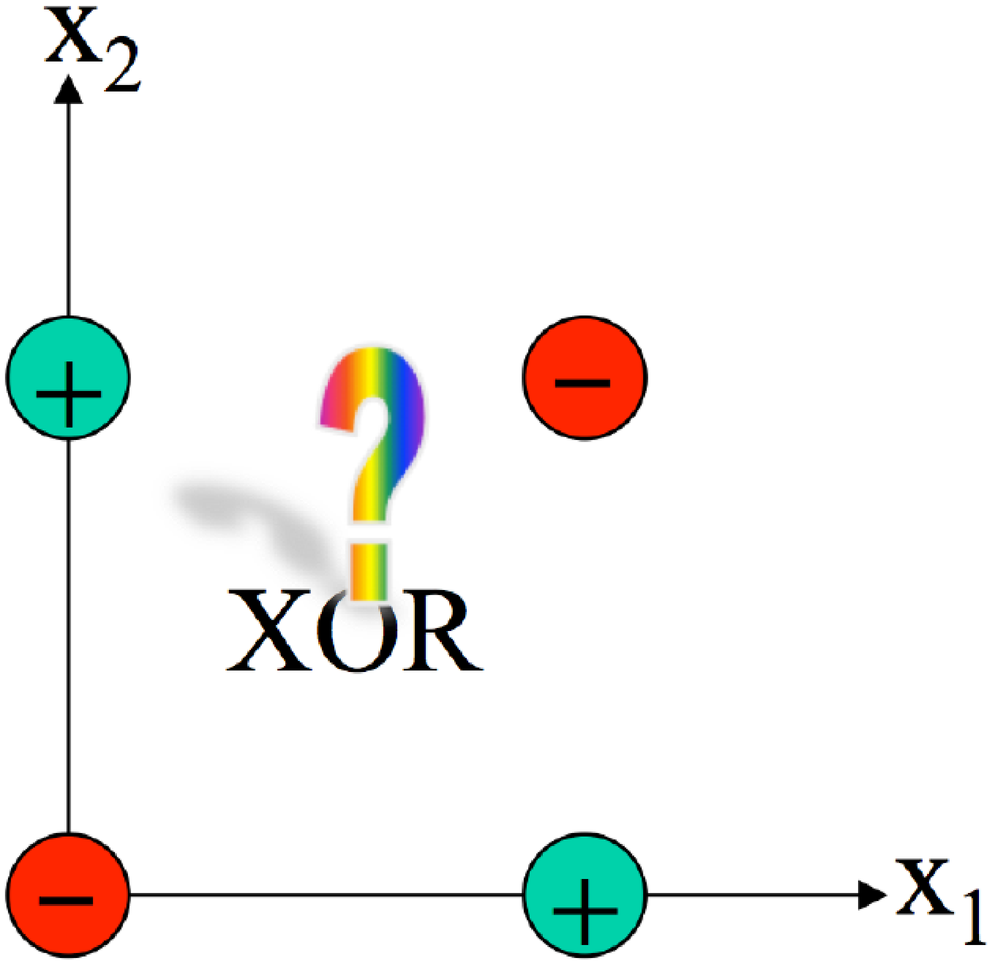
p=3

……

What about XOR?



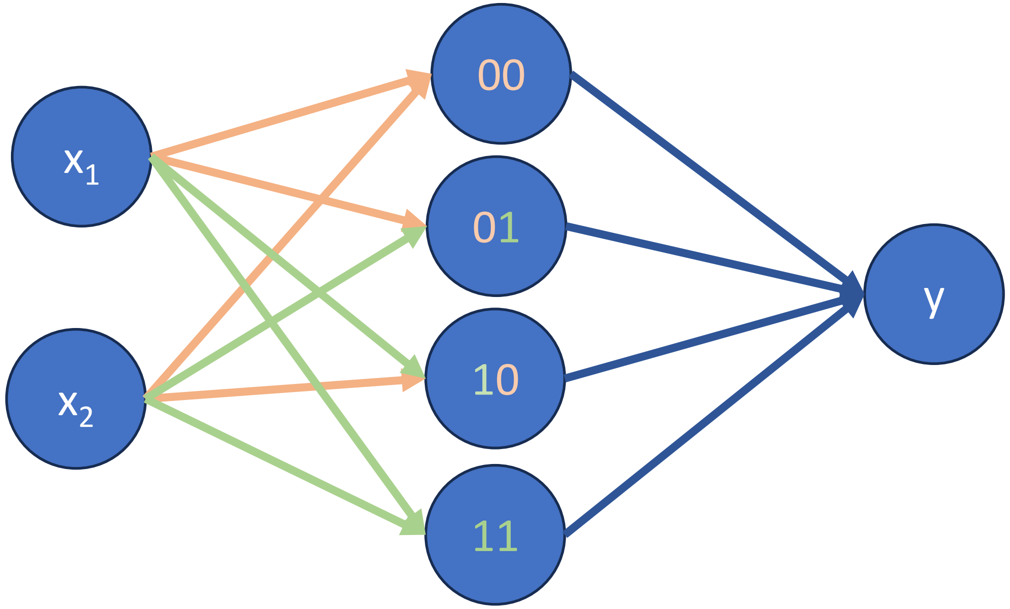
What about XOR?



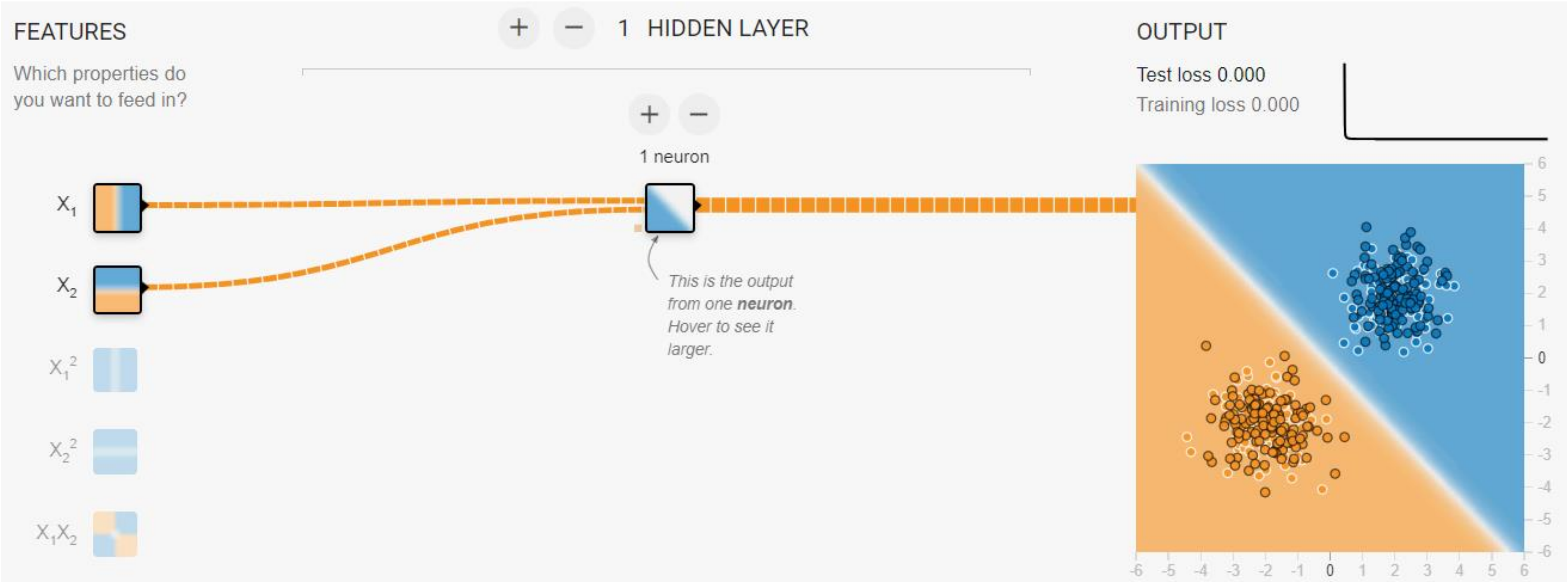
One single perceptron can not solve the problem.
Need multiple perceptron in a layered architecture.

Representation Power of NNs

- NN with 1 hidden layer can represent:
  - Any bounded continuous function (to arbitrary ε)
    - Universal Approximation Theorem [Cybenko 1989]
  - Any Boolean function (exactly)

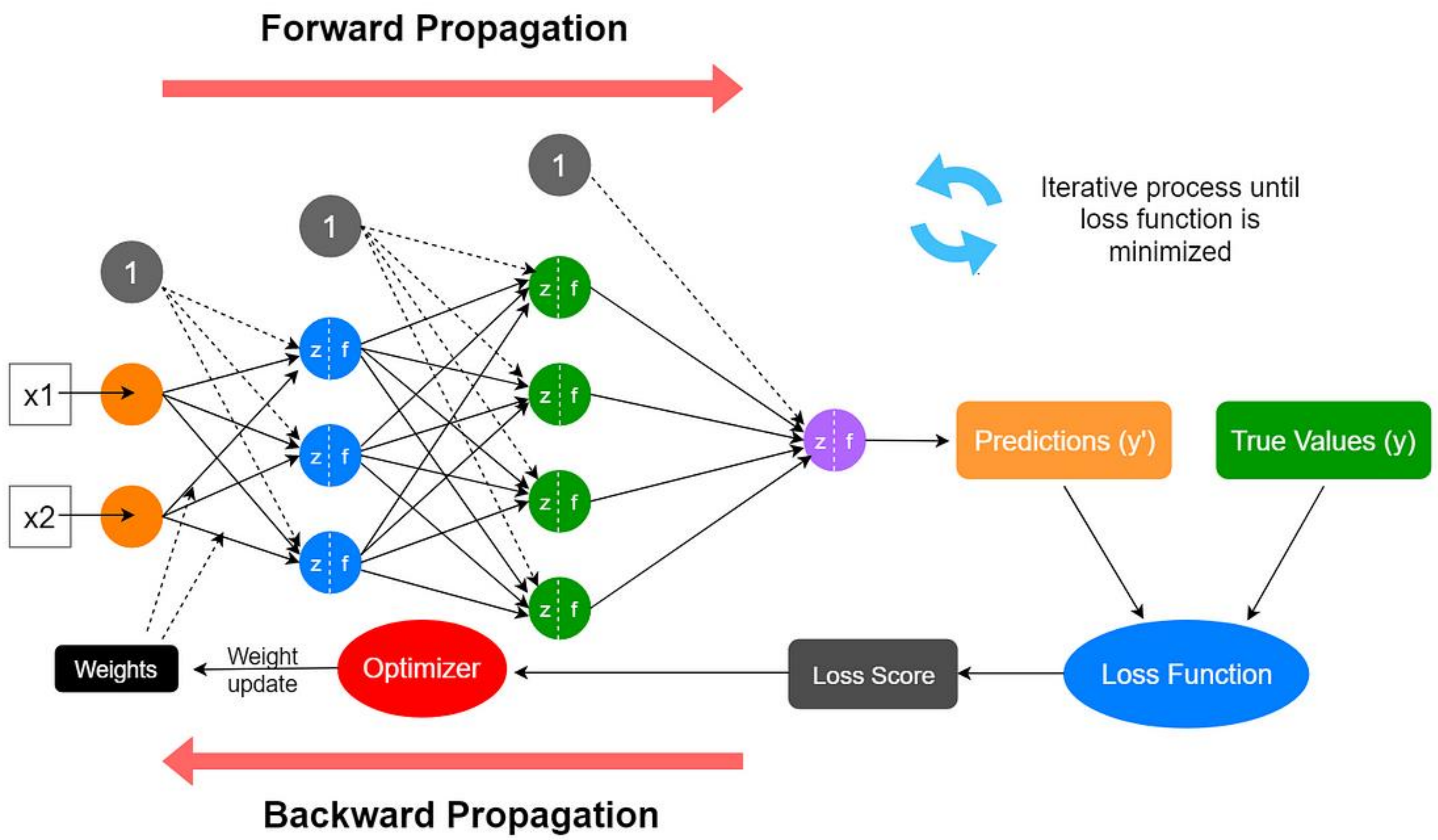


Tensorflow playground



https://playground.tensorflow.org/

Neural Network Learning Process



Forward step

In terms of indexes introduce the following:

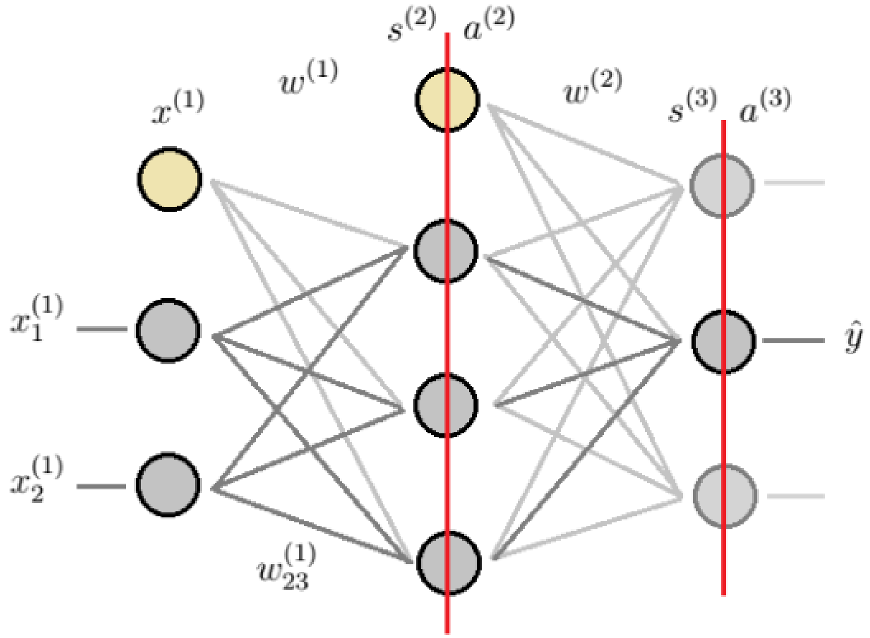
$a_i^{(j)}$ - activation of the i^{th} neuron in the j^{th} layer

$s_i^{(j)}$ - summed output of the i^{th} neuron in the j^{th} layer

$w_{lk}^{(j)}$ - weight between the l^{th} neuron in the j^{th} layer and the k^{th} neuron in the $j+1^{th}$ layer

$x_n^{(m)}$ - the n^{th} feauture in the m^{th} input. ($x_0^{(1)} = 1$) is the BIAS.

$\hat{y}$ - the output



Forward step

The BIAS is added to the $x^{(1)}$ vector and multiplied by the first weight matrix.

$$\underset{1\times3}{\mathrm{x}^{(1)}} \times \underset{3\times3}{\mathrm{w}^{(1)}} = \underset{1\times3}{\mathrm{s}^{(2)}}$$

We perform the activation in the neurons of the hidden layer. We use the sigmoid function as activation function.

$$\underset{1\times3}{\mathrm{a}^{(2)}} = f(\underset{1\times3}{\mathrm{s}^{(2)}}) = sigmoid(\underset{1\times3}{\mathrm{s}^{(2)}})$$

We assign the BIAS to the hidden layer after the activation, but before the weight is applied!

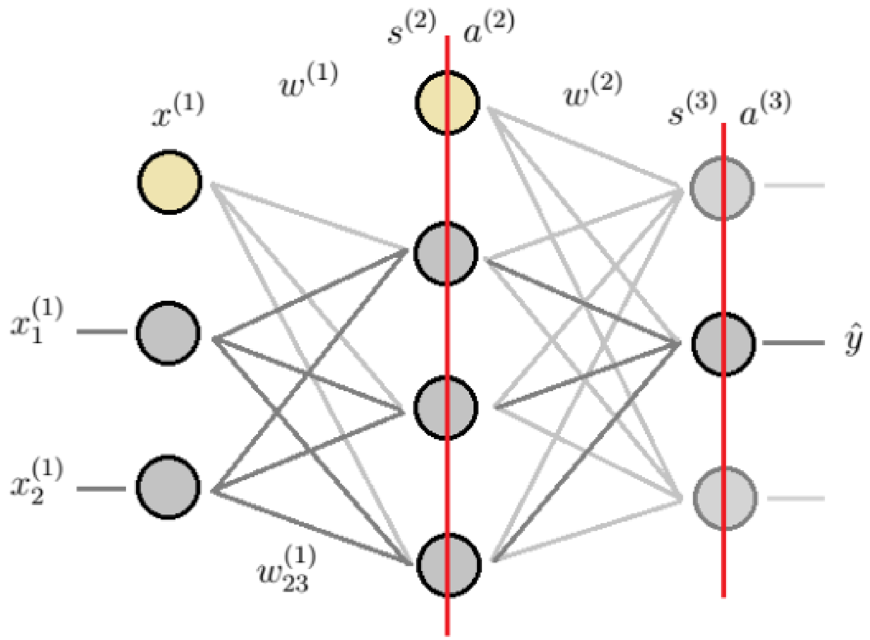
$$\underset{1\times4}{\mathrm{a}^{(2)}} \times \underset{4\times3}{\mathrm{w}^{(2)}} = \underset{1\times3}{\mathrm{s}^{(3)}}$$

After activation the value of the neurons in the output layer is obtained.

$$\underset{1\times3}{\mathrm{a}^{(3)}} = f(\underset{1\times3}{\mathrm{s}^{(3)}}) = sigmoid(\underset{1\times3}{\mathrm{s}^{(3)}})$$

The output layer contains the predictions.

$$\underset{1\times3}{\hat{\mathrm{y}}} = \underset{1\times3}{\mathrm{a}^{(3)}}$$



Cost function

$$C = \sum\{\tfrac{1}{2}(y - \hat{y})^2\}$$

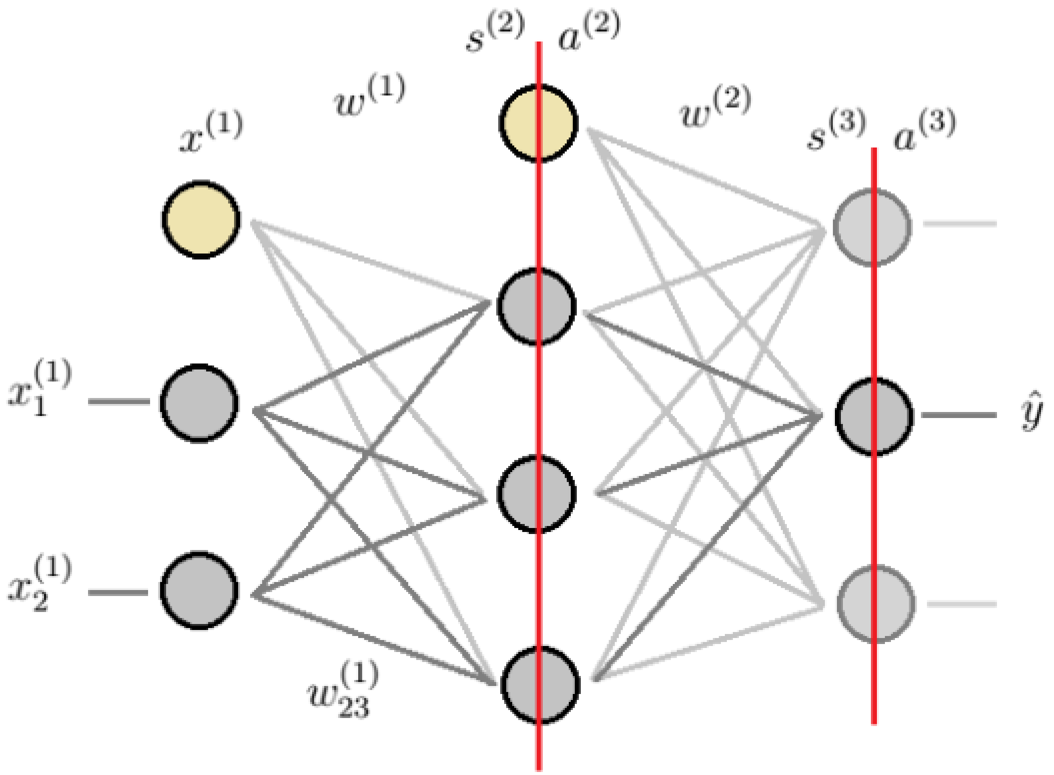
$$C = \sum\{\tfrac{1}{2}(y - a^{(3)})^2\}$$

$$C = \sum\{\tfrac{1}{2}(y - f(s^{(3)}))^2\}$$

$$C = \sum\{\tfrac{1}{2}(y - f(a^{(2)}w^{(2)}))^2\}$$

$$C = \sum\{\tfrac{1}{2}(y - f(f(s^{(2)})w^{(2)}))^2\}$$

$$C = \sum\{\tfrac{1}{2}(y - f(f(xw^{(1)})w^{(2)}))^2\}$$

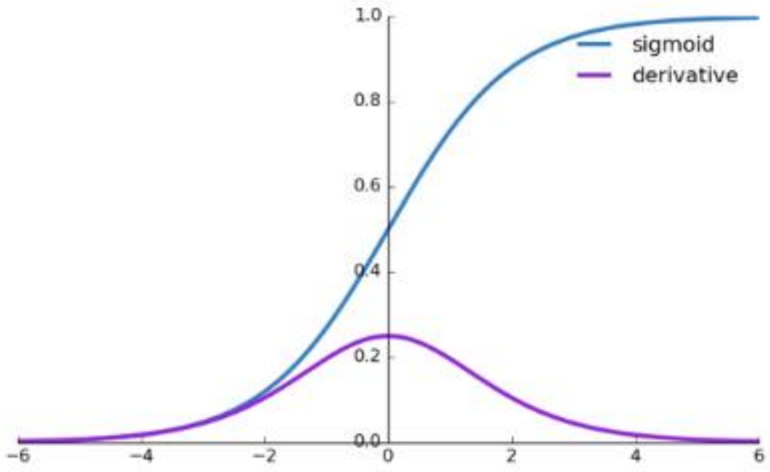


Back propagation

Sigmoid function and its derivative

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

$$\begin{aligned} g'(z) &= = \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}(e^{(-z)}) \\ &= \frac{1}{1+e^{-z}}(1 - \frac{1}{(1+e^{-z})}) \\ &= g(z)(1-g(z)) \end{aligned}$$

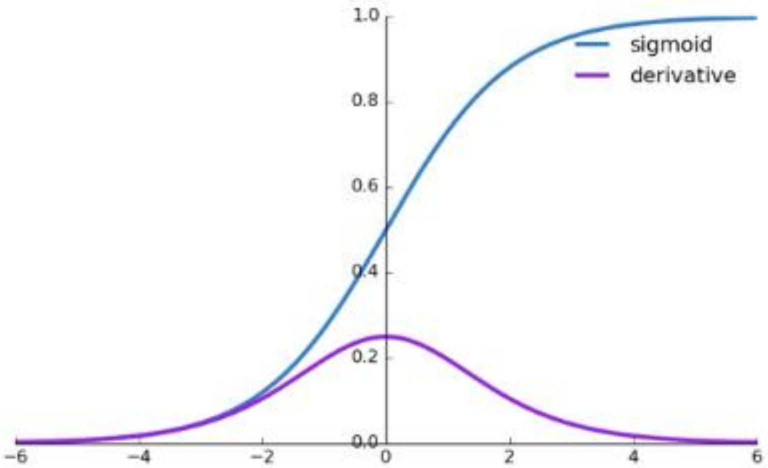


Back propagation

Sigmoid function and its derivative

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

$$g'(z) = = \frac{d}{dz}\frac{1}{1+e^{-z}}$$
$$= \frac{1}{(1+e^{-z})^2}(e^{(-z)})$$
$$= \frac{1}{1+e^{-z}}(1 - \frac{1}{(1+e^{-z})})$$
$$= g(z)(1-g(z))$$



$$\frac{\partial C}{\partial w^{(2)}} = \frac{\partial \sum \frac{1}{2}(y-\hat{y})^2}{\partial w^{(2)}} = \sum(\frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial w^{(2)}})$$

For the sake of clarity, let's derive the derivation for one element.

$$\frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial w^{(2)}} = (y-\hat{y})(-\frac{\hat{y}}{\partial w^{(2)}})$$
$$= -(y-\hat{y}) \cdot \frac{\partial \hat{y}}{\partial s^{(3)}} \cdot \frac{\partial s^{(3)}}{\partial w^{(2)}}$$
$$= -(y-\hat{y}) \cdot f'(s^{(3)}) \cdot \frac{\partial a^{(2)} w^{(2)}}{\partial w^{(2)}}$$
$$= \delta^{(3)} \cdot a^{(2)}$$

Introduce $\delta_i^{(j)}$ as the error term assigned to neuron i of layer j.

Extended in matrix form, after checking the dimensions, we obtain the following relation:

$(a^{(2)})^T \delta^{(3)}$

Full algorithm

1, $xw^{(1)} = s^{(2)}$

2, $f(s^{(2)}) = a^{(2)}$

3, $a^{(2)}w^{(2)} = s^{(3)}$

4, $f(s^{(3)}) = \hat{y}$

5, $C = \sum\{\frac{1}{2}(y - \hat{y})^2\}$

6, $-(y - \hat{y}) \cdot f'(s^{(3)}) = \delta^{(3)}$

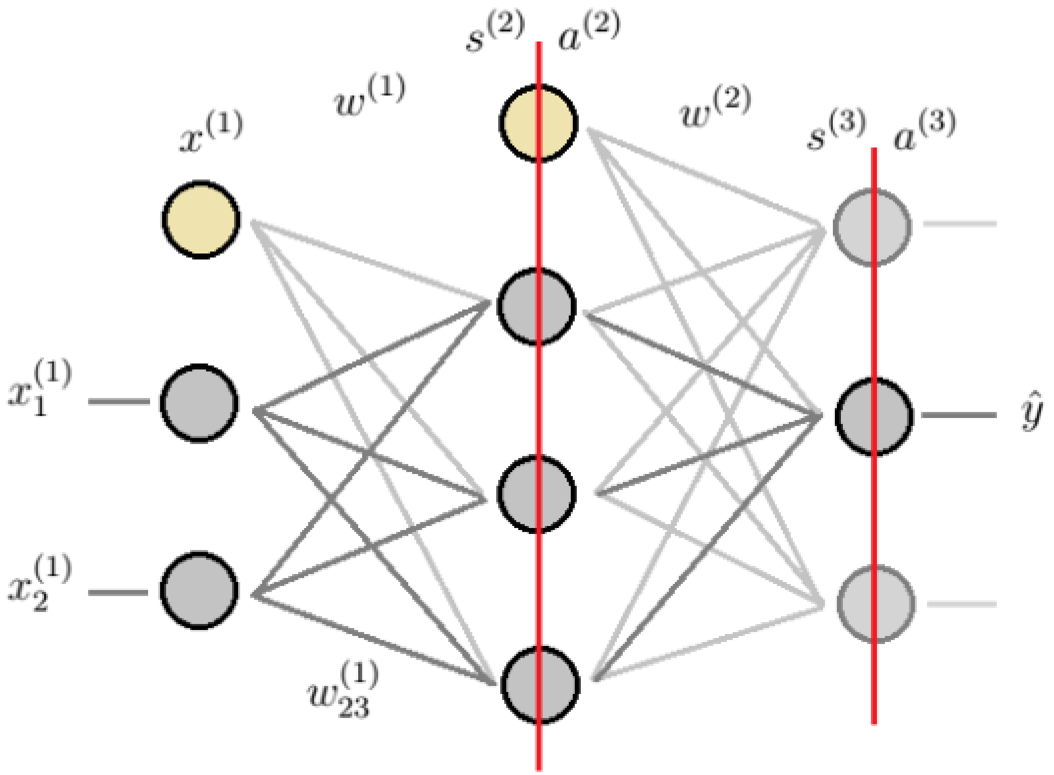
7, $(a^{(2)})^T \delta^{(3)} = \frac{\partial C}{\partial w^{(2)}}$

8, $\delta^{(3)} \cdot (w^{(2)})^T \cdot f'(s^{(2)}) = \delta^{(2)}$

9, $x^T \delta^{(2)} = \frac{\partial C}{\partial w^{(1)}}$

10, $w^{(1)} = w^{(1)} - \mu \frac{\partial C}{w^{(1)}} + regularization$

$w^{(2)} = w^{(2)} - \mu \frac{\partial C}{w^{(2)}} + regularization$



FACULTY OF INFORMATICS

Thank you for your attention!