Machine Learning Journal Club

Lecture 2

A journey into neural networks:

from perceptrons to CNNs

WHAT
MACHINE LEARNING
REALLY MEANS

Components of learning

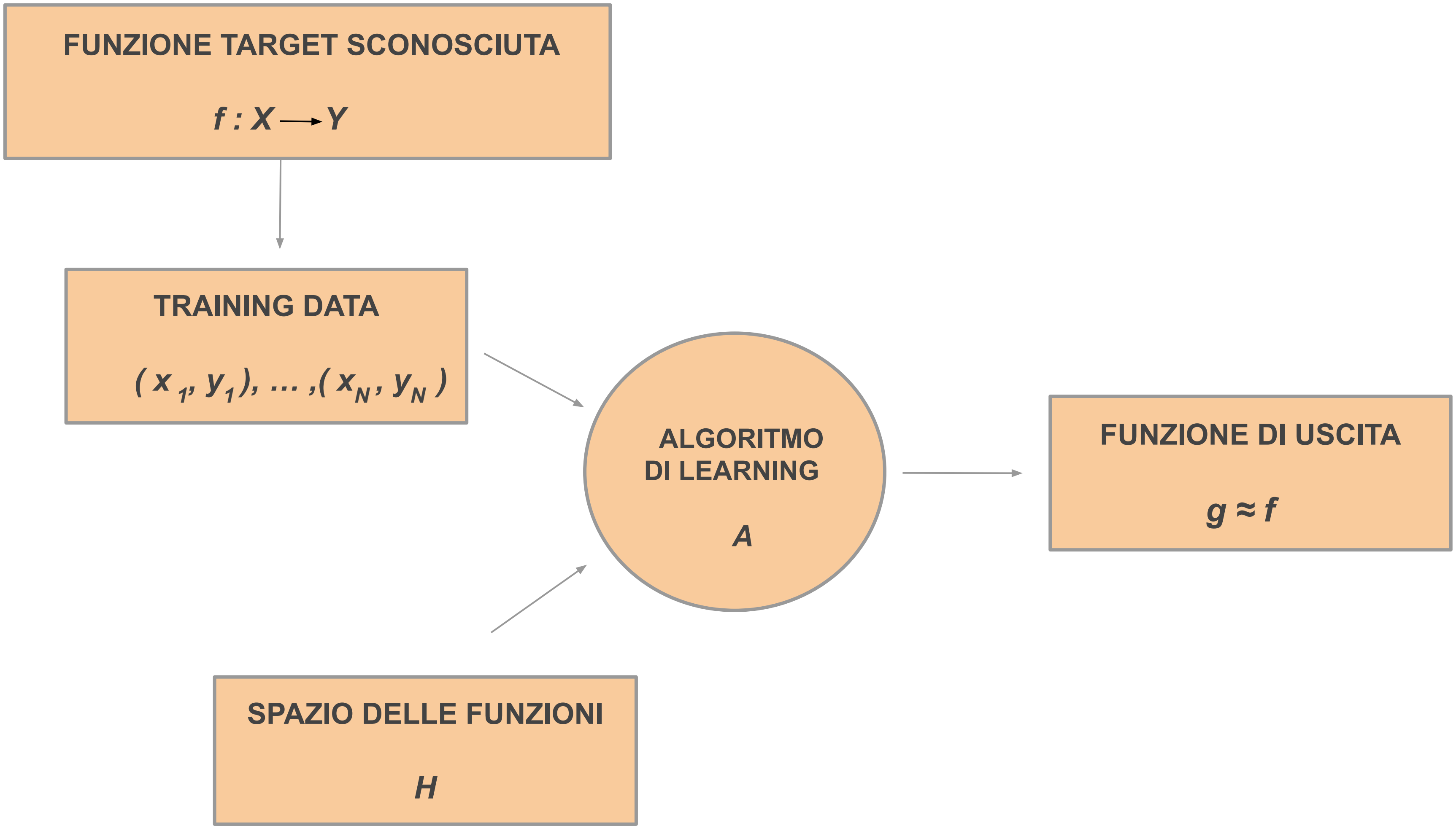
- *Input:* $\mathbf{x}$
- *Output:* y
- *Target:* $\mathbf{f}: X \rightarrow Y$

- *Data:* $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2) \dots (\mathbf{x_N}, y_N)$

↓ ↓

- *Hypothesis:* $\mathbf{g}: X \rightarrow Y$

A



FUNZIONE TARGET SCONOSCIUTA

$f : X \longrightarrow Y$

TRAINING DATA

$(x_1, y_1), \ldots ,(x_N, y_N)$

ALGORITMO
DI LEARNING

A

FUNZIONE DI USCITA

$g \approx f$

SPAZIO DELLE FUNZIONI

H

A Simple learning model - the perceptron

- Let $X = R^d$ be the input space, so $x = (x_1, \dots, x_d)$
- Let $Y = \{+1,-1\}$ be the output space (binary decision)

$$\text{Approve if } \sum_{i=1}^{d} w_i x_i > \text{threshold}$$

$$\text{Deny if } \sum_{i=1}^{d} w_i x_i < \text{threshold}$$

$$\longrightarrow \quad h(\boldsymbol{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{treshold}\right)$$

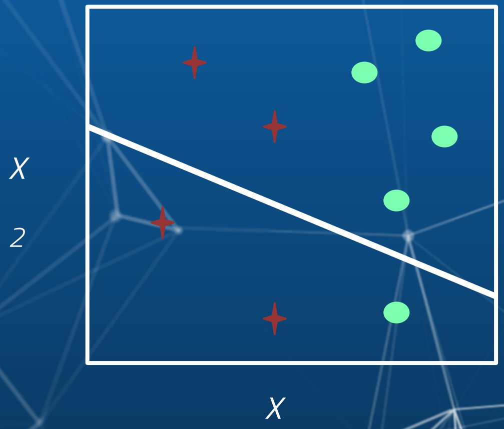
A Simple learning model - the perceptron

$$h(\boldsymbol{x})=sign((\sum_{i=1}^{d} w_i x_i) + w_0)$$

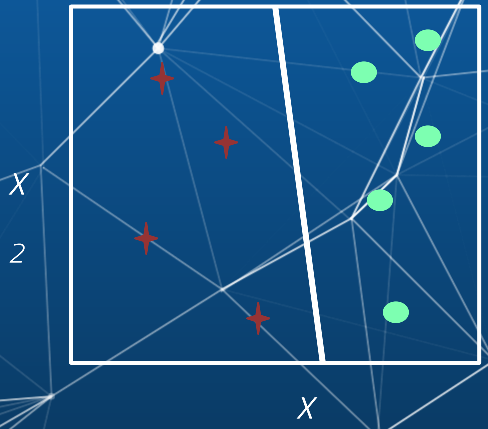
$$h(\boldsymbol{x})=sign(\sum_{i=0}^{d} w_i x_i)$$

$$h(\boldsymbol{x})=sign(\boldsymbol{w}^T\boldsymbol{x})$$

2d - perceptron



Misclassified data



Perfectly classified data

Perceptron Learning Algorithm (PLA)

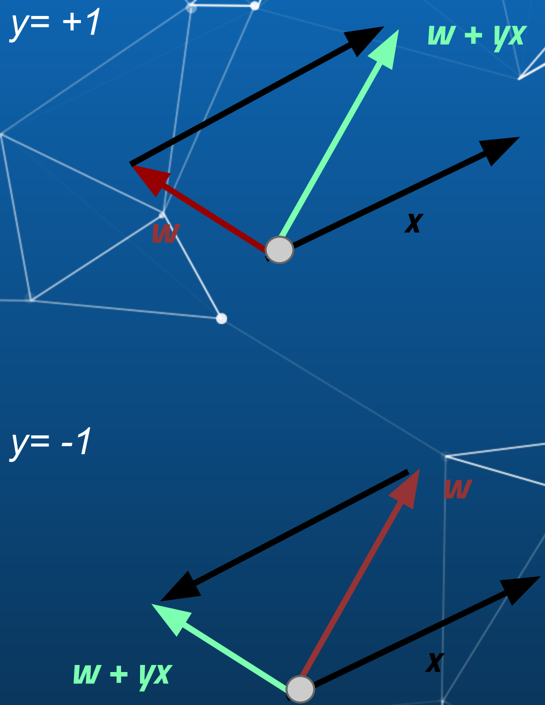
- ❑ *Given the training set : ($\mathbf{x}_1$, y_1),…,($\mathbf{x}_N$, y_N)*

- ➢ *Pick a misclassified point*

$$sign(w^T x_n) \neq y_n$$

- ➢ *update the weight vector*

$$w \leftarrow w + x_n y_n$$

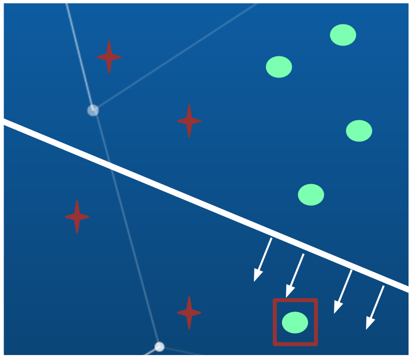


Iterations of PLA

- ☐ *Call w(t) the current weight vector at iteration t, t=0,1,2…*

- ☐ *Pick a currently misclassified example (x(t),y(t)) and update weights:*

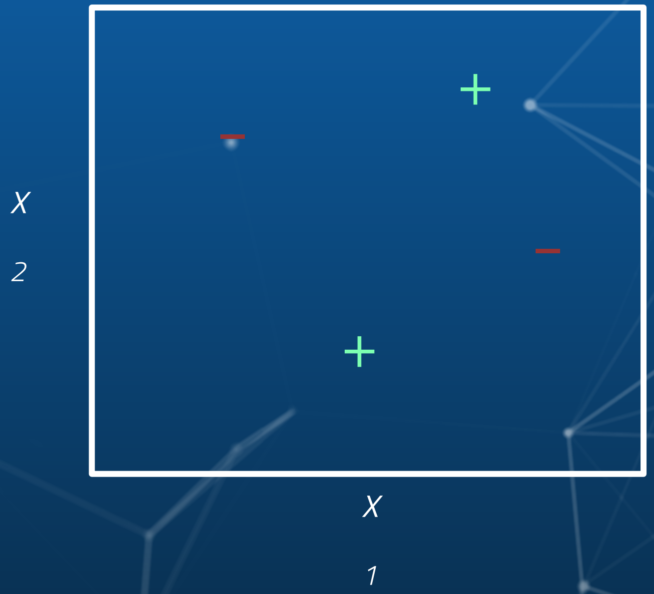
$$w(t+1) = w(t) + y(t)x(t)$$



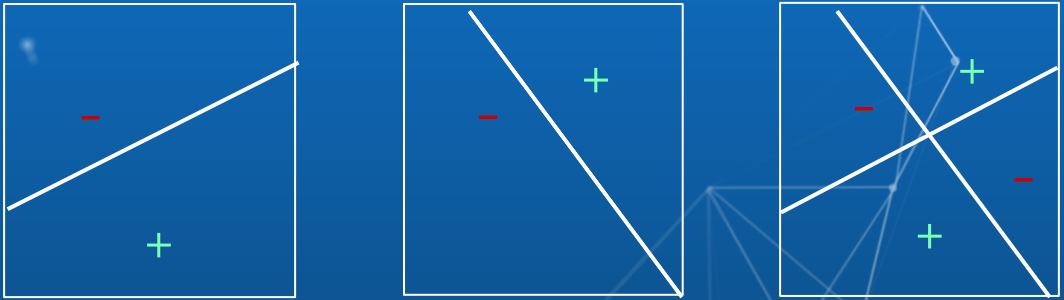
- ☐ *The iteration goes on untill there are no more misclassified points!*

The Multy-layer perceptron (MLP)

- *Take a look to the figure below:*
  The target function cannot be written as $sign(w^T x)$



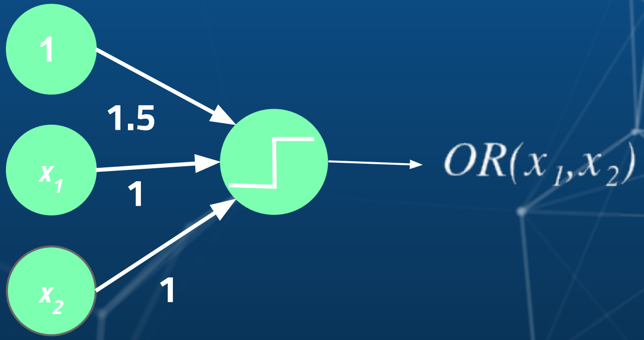
The Multy-layer perceptron (MLP)



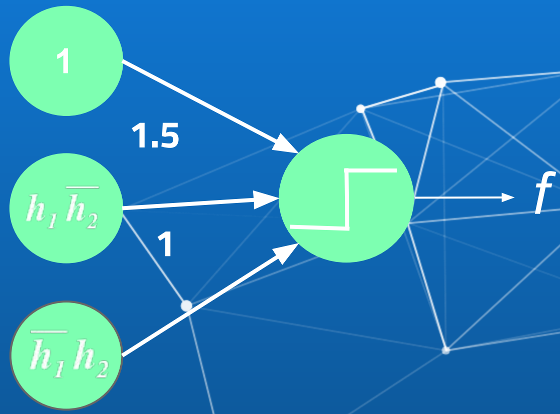
$h_1(\mathbf{x})=sign(\mathbf{w}_1^T\mathbf{x})$ $h_2(\mathbf{x})=sign(\mathbf{w}_2^T\mathbf{x})$ $f=h_1\overline{h_2}+\overline{h_1}h_2$

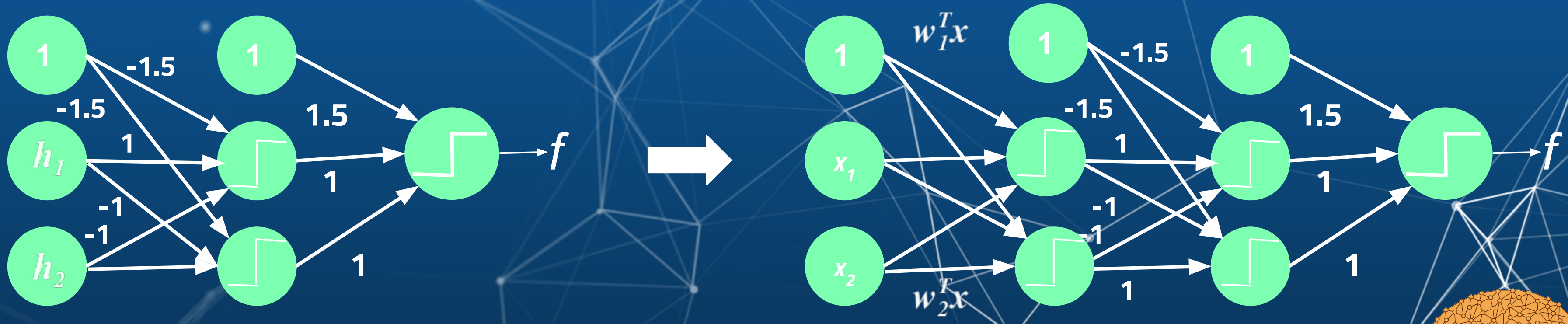
- ❑ $f = XOR(h_1,h_2)$

0	0	0
0	1	1
1	0	1
1	1	0

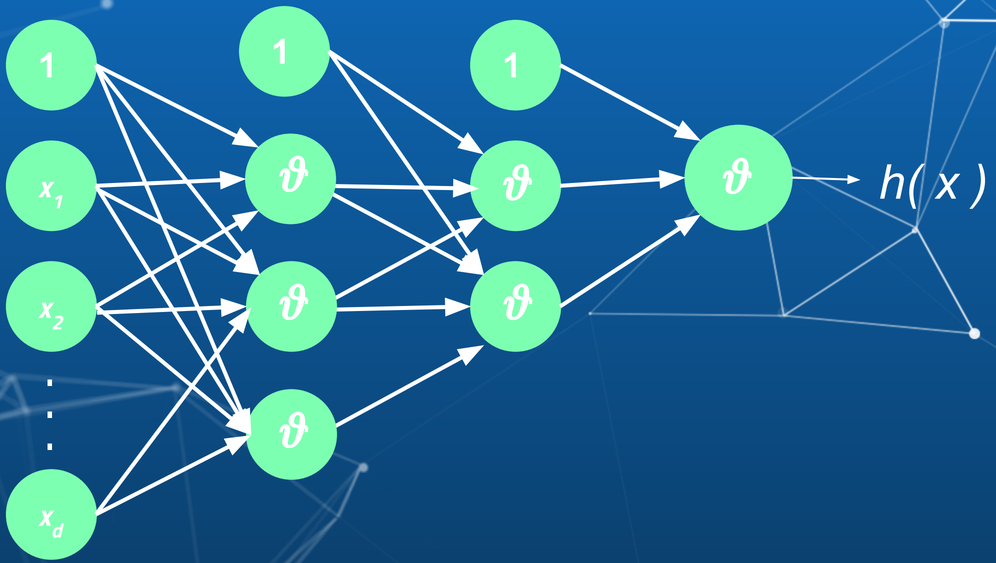


Since $f = h_1\overline{h_2} + \overline{h_1}h_2$, which is an OR of the two inputs $h_1\overline{h_2}$ and $\overline{h_1}h_2$

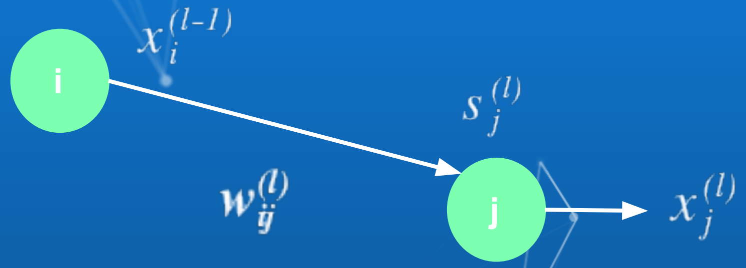




Neural Networks



Input layer l=0 *hidden layers 0< l < L* *output layer l = L*



Layer l-1 **Layer l**

$x_i^{(l-1)}$: *output node i, related to layer l-1*

$w_{ij}^{(l)}$: *elements of weights matrix that goes from i to j*

$s_j^{(l)}$: *Signal transferred to node j*

$\theta(s_j^l)$: *Activation function for every signal* s_j

$$\boxed{s_j^{(l)} = \sum_{i=0}^{d} w_{ij}^{l} x_i^{l-1}}$$

$$\boxed{\theta(s_j^l) = x_j^l}$$

Activation Functions

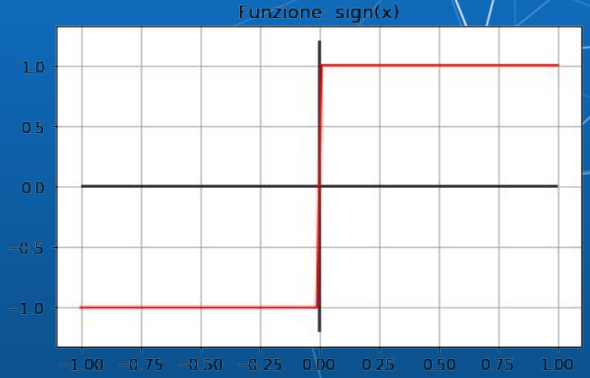
The choice of the activation function is arbitrary:

- ***Hard Threshold:***

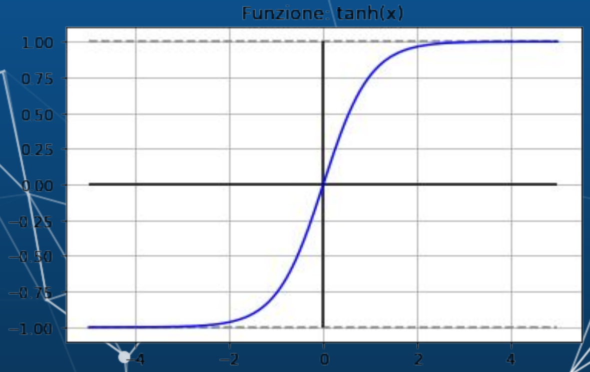
$$\theta(s) = sign(s)$$

- ***Soft Threshold***

$$\theta(s) = tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$



Funzione: sign(x)



Funzione: tanh(x)

Forward propagation

- *In order to compute h(x) we use the forward propagation algorithm*

- *We have already seen that* $\boxed{x^{(l)} = \frac{1}{\theta(s^{(l)})}}$ *and* $\boxed{s^{(l)} = (W^{(l)})^T x^{(l-1)}}$

- *So we initialize the input layer to* $x^{(0)} = x$ *and compute:*

$$x = x^{(0)} \xrightarrow{W^1} s^{(1)} \xrightarrow{\theta} x^{(1)} \xrightarrow{W^2} \dots \rightarrow s^{(L)} \xrightarrow{\theta} x^{(L)} = h(x)$$

The Error function

- ❑ *The training is made when we update the weight vector* ***w*** *, minimizing the error function*.

$$E_{in} = \frac{1}{N} \sum_{i=1}^{N} e_n$$

- ❑ We define e_n depending on the problem :

  - ➢ $e_n(w) = || h(x_n) \neq y_n ||$ *Classification*
  - ➢ $e_n(w) = (h(x_n) - y_n)^2$ *Other problems*

- How can we minimize $E_{in}(w)$?

↓

Gradient descent:

$$\boxed{\mathbf{w}(t+1)=\mathbf{w}(t)-\eta\nabla E_{in}(\mathbf{w}(t))}$$

with $\mathbf{w}=[W^1,W^2\dots W^L]$

- So we need the $E_{in}(w)$ derivative respect to each weight matrix:

$$\frac{\partial E_{in}}{\partial W^{(l)}}=\frac{1}{N}\sum_{n=1}^{N}\frac{\partial e_n}{\partial W^{(l)}}$$

Back Propagation

- Let define the **sensitivity**:

$$\delta_j^{(l)} = \frac{\partial e_n(w)}{\partial s_j^{(l)}} = \frac{\partial e_n(w)}{\partial x_j^{(l)}} \frac{\partial x_j^{(l)}}{\partial s_j^{(l)}}$$

$$\frac{\partial e_n(w)}{\partial x_j^{(l)}} = \sum_{k=1}^{d^{(l+1)}} \frac{\partial s_k^{(l+1)}}{\partial x_j^{(l)}} \frac{\partial e_n(w)}{\partial s_k^{(l+1)}} = \sum_{k=1}^{d^{(l+1)}} w_{jk}^{(l+1)} \delta_k^{(l+1)}$$

$$\delta_j^{(l)} = \theta'(s_j^{(l)}) \sum_{k=1}^{d^{(l+1)}} w_{jk}^{(l+1)} \delta_k^{(l+1)}$$

Back Propagation Algorithm

- *Training can be achieved since:*

$$\delta^{(L)} = \frac{\partial e}{\partial s^{(L)}} = 2(x^{(L)} - y)\frac{\partial x^{(L)}}{\partial s^{(L)}} = 2(x^{(L)} - y)\theta'(s^{(L)})$$

- *In this way we can evaluate all the sensitivities:*

$$\delta^{(1)} \leftarrow \delta^{(2)} \leftarrow \ldots \leftarrow \delta^{(L)}$$

So we can compute:

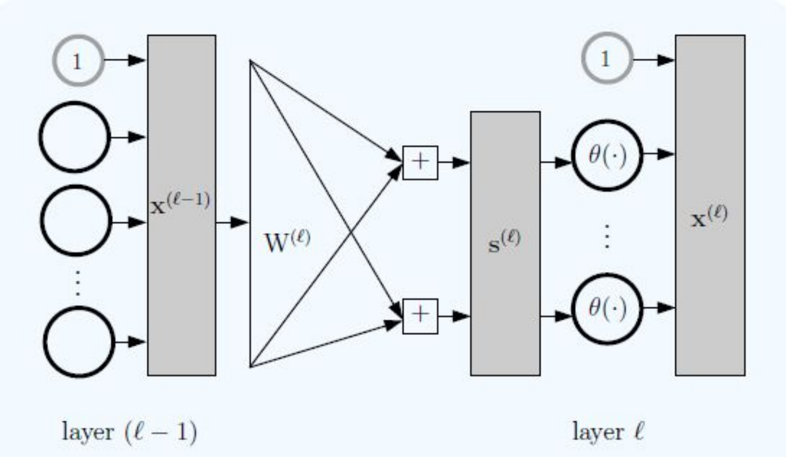
$$\frac{\partial e}{\partial w_{ij}^{(l)}} = \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial e}{\partial s_j^{(l)}} = x_i^{(l-1)} \delta_j^{(l)}$$

$$\downarrow$$

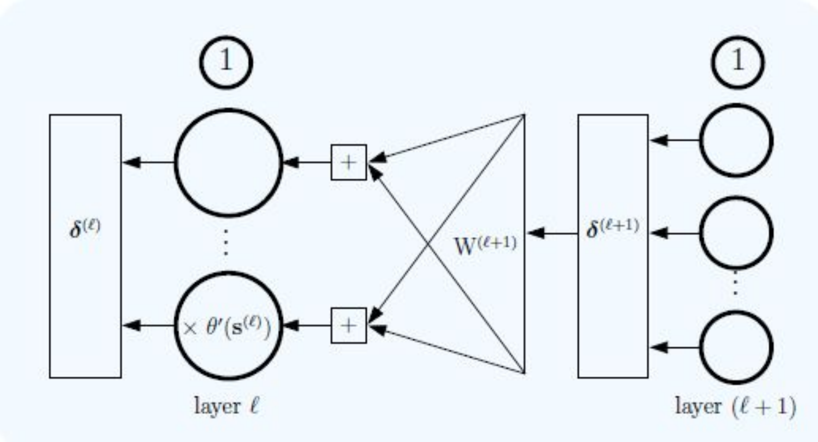
$$G^{(l)}(\mathbf{x}_n) = \frac{\partial e_n}{\partial W^{(l)}} = [\mathbf{x}^{(l-1)}(\delta^{(l)})^T]$$

Summary

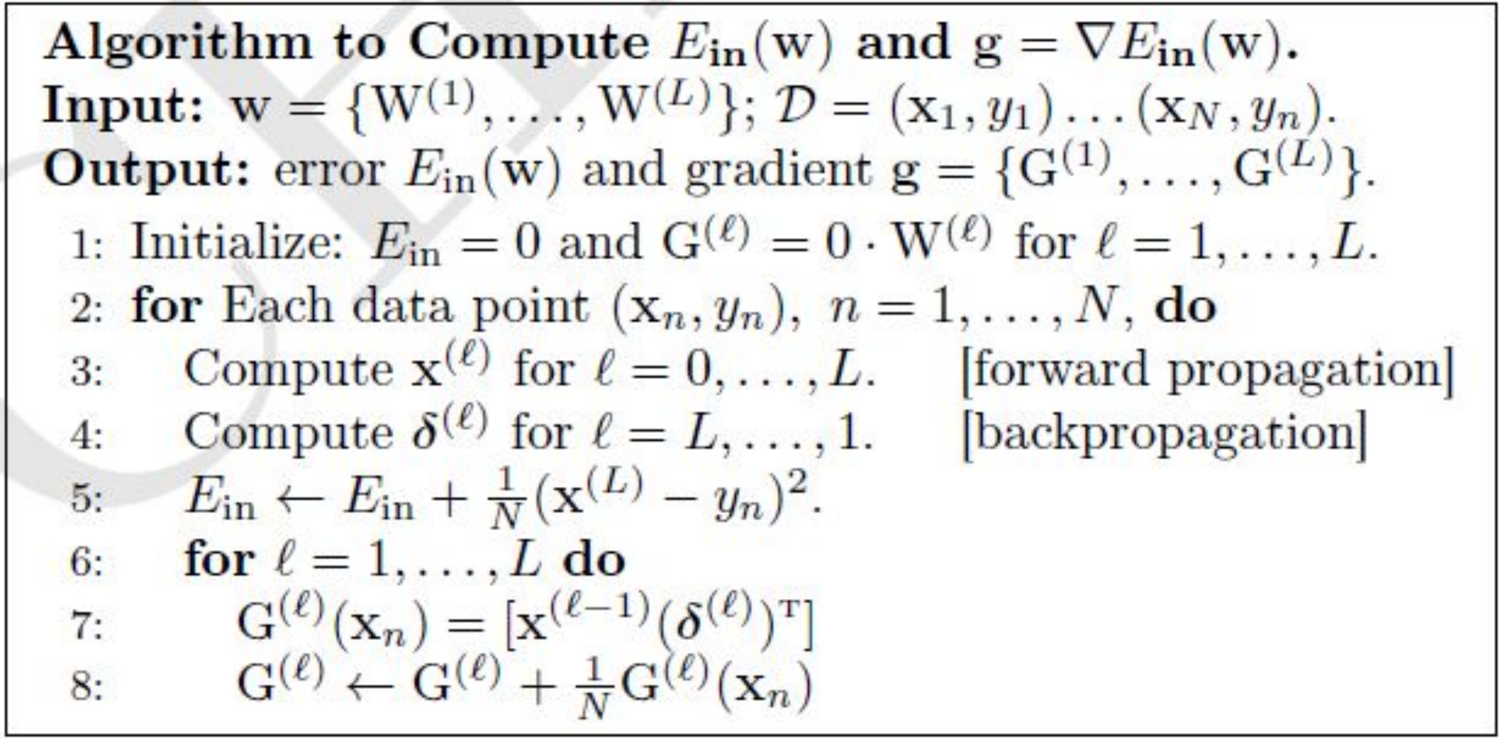
Forward propagation



back propagation



Back Propagation Algorithm



```
**Algorithm to Compute** $E_{\text{in}}(\mathrm{w})$ **and** $\mathrm{g} = \nabla E_{\text{in}}(\mathrm{w})$.
**Input:** $\mathrm{w} = \{\mathrm{W}^{(1)}, \ldots, \mathrm{W}^{(L)}\}; \mathcal{D} = (\mathrm{x}_1, y_1) \ldots (\mathrm{x}_N, y_n)$.
**Output:** error $E_{\text{in}}(\mathrm{w})$ and gradient $\mathrm{g} = \{\mathrm{G}^{(1)}, \ldots, \mathrm{G}^{(L)}\}$.
Initialize: $E_{\text{in}} = 0$ and $\mathrm{G}^{(\ell)} = 0 \cdot \mathrm{W}^{(\ell)}$ for $\ell = 1, \ldots, L$.
for Each data point $(\mathrm{x}_n, y_n)$, $n = 1, \ldots, N$, do
    Compute $\mathrm{x}^{(\ell)}$ for $\ell = 0, \ldots, L$.    [forward propagation]
    Compute $\delta^{(\ell)}$ for $\ell = L, \ldots, 1$.    [backpropagation]
    $E_{\text{in}} \leftarrow E_{\text{in}} + \frac{1}{N}(\mathrm{x}^{(L)} - y_n)^2$.
    for $\ell = 1, \ldots, L$ do
        $\mathrm{G}^{(\ell)}(\mathrm{x}_n) = [\mathrm{x}^{(\ell-1)}(\delta^{(\ell)})^{\mathrm{T}}]$
        $\mathrm{G}^{(\ell)} \leftarrow \mathrm{G}^{(\ell)} + \frac{1}{N}\mathrm{G}^{(\ell)}(\mathrm{x}_n)$
```