

Learning From Data

Lecture 21

Neural Networks: Fitting the Data

Forward propagation: algorithmic computation $h(\mathbf{x})$

Backpropagation: algorithmic computation of $\frac{\partial e(\mathbf{x})}{\partial \text{weights}}$

Fitting or Overfitting?

M. Magdon-Ismail

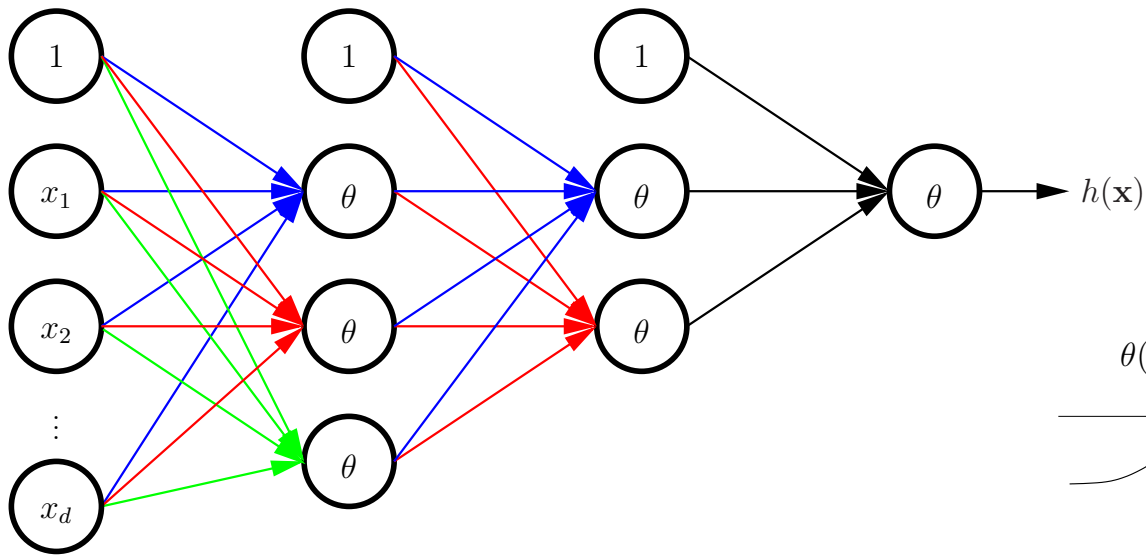
CSCI 4100/6100

RECAP: **The Neural Network**



Biology →

Engineering ↓

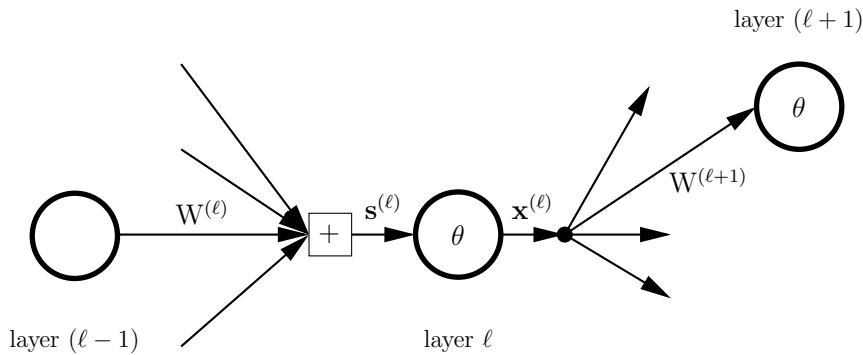
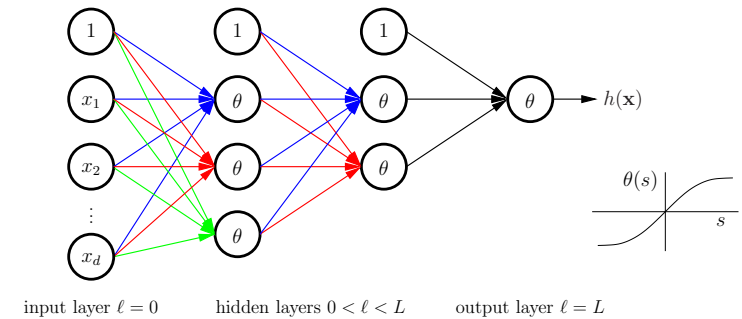


input layer $\ell = 0$

hidden layers $0 < \ell < L$

output layer $\ell = L$

Zooming into a Hidden Node



layer ℓ parameters

signals in	$\mathbf{s}^{(\ell)}$	$d^{(\ell)}$ dimensional input vector
outputs	$\mathbf{x}^{(\ell)}$	$d^{(\ell)} + 1$ dimensional output vector
weights in	$\mathbf{W}^{(\ell)}$	$(d^{(\ell-1)} + 1) \times d^{(\ell)}$ dimensional matrix
weights out	$\mathbf{W}^{(\ell+1)}$	$(d^{(\ell)} + 1) \times d^{(\ell+1)}$ dimensional matrix

layers $\ell = 0, 1, 2, \dots, L$
 layer ℓ has “dimension” $d^{(\ell)} \implies d^{(\ell)} + 1$ nodes

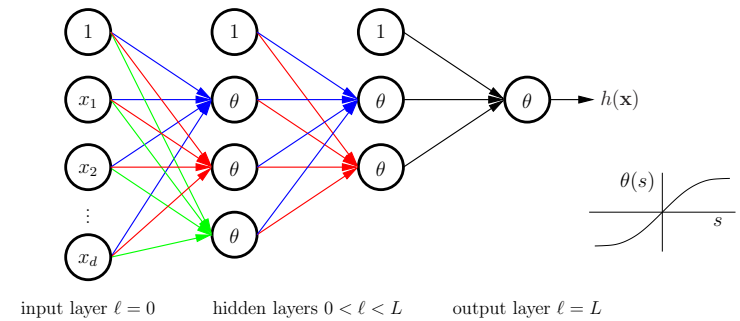
$$\mathbf{W}^{(\ell)} = \begin{bmatrix} \mathbf{w}_1^{(\ell)} & \mathbf{w}_2^{(\ell)} & \cdots & \mathbf{w}_{d^{(\ell)}}^{(\ell)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L)}\}$ \leftarrow specifies the network

The Linear Signal

Input $\mathbf{s}^{(\ell)}$ is a linear combination (using weights) of the outputs of the previous layer $\mathbf{x}^{(\ell-1)}$.

$$\mathbf{s}^{(\ell)} = (\mathbf{W}^{(\ell)})^T \mathbf{x}^{(\ell-1)}$$



$$\begin{bmatrix} s_1^{(\ell)} \\ s_2^{(\ell)} \\ \vdots \\ s_j^{(\ell)} \\ \vdots \\ s_{d^{(\ell)}}^{(\ell)} \end{bmatrix} = \begin{bmatrix} (\mathbf{w}_1^{(\ell)})^T \text{-----} \\ (\mathbf{w}_2^{(\ell)})^T \text{-----} \\ \vdots \\ (\mathbf{w}_j^{(\ell)})^T \text{-----} \\ \vdots \\ (\mathbf{w}_{d^{(\ell)}}^{(\ell)})^T \text{-----} \end{bmatrix} \mathbf{x}^{(\ell-1)}$$

$$s_j^{(\ell)} = (\mathbf{w}_j^{(\ell)})^T \mathbf{x}^{(\ell-1)}$$

(recall the linear signal $s = \mathbf{w}^T \mathbf{x}$)

$$\mathbf{s}^{(\ell)} \xrightarrow{\theta} \mathbf{x}^{(\ell)}$$

Forward Propagation: Computing $h(\mathbf{x})$

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{w^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{w^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \dots \xrightarrow{w^{(L)}} \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x}).$$

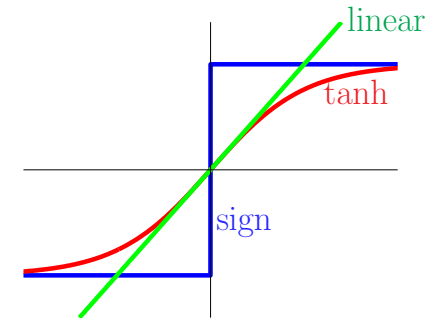
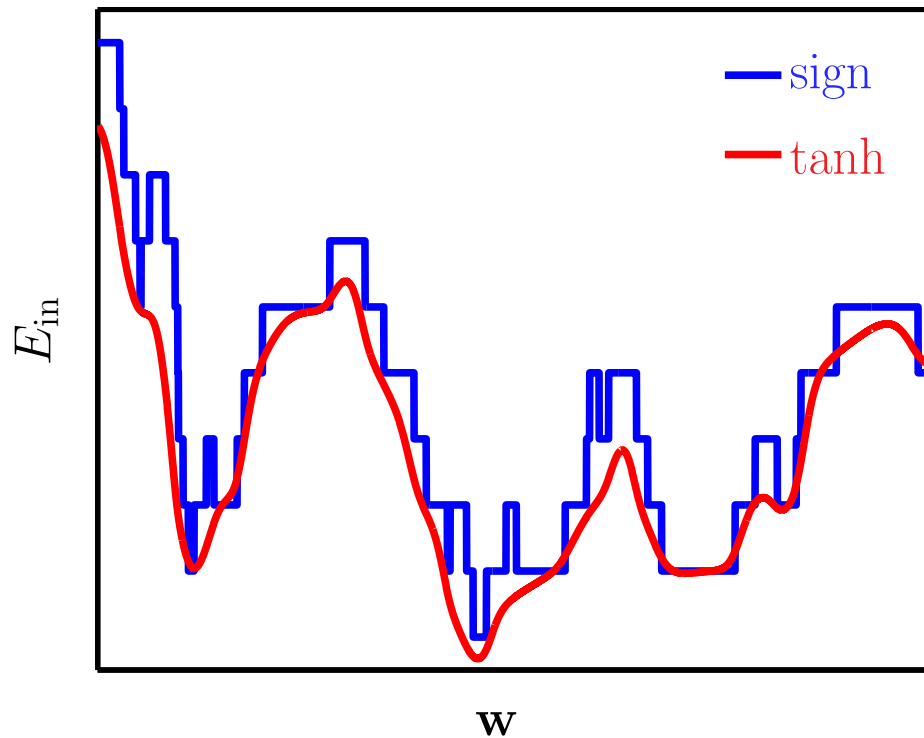
Forward propagation to compute $h(\mathbf{x})$:

- 1: $\mathbf{x}^{(0)} \leftarrow \mathbf{x}$ [Initialization]
- 2: **for** $\ell = 1$ to L **do** [Forward Propagation]
- 3: $\mathbf{s}^{(\ell)} \leftarrow (\mathbf{W}^{(\ell)})^T \mathbf{x}^{(\ell-1)}$
- 4: $\mathbf{x}^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(\mathbf{s}^{(\ell)}) \end{bmatrix}$
- 5: **end for**
- 6: $h(\mathbf{x}) = \mathbf{x}^{(L)}$ [Output]

Minimizing E_{in}

$$E_{\text{in}}(h) = E_{\text{in}}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^N (h(\mathbf{x}_n) - y_n)^2$$

$$\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L)}\}$$



Using $\theta = \text{tanh}$ makes E_{in} differentiable so we can use gradient descent \rightarrow local minimum.

Gradient Descent

$$W(t + 1) = W(t) - \eta \nabla E_{\text{in}}(W(t))$$

Gradient of E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \overset{e_n}{\mathbf{e}(h(\mathbf{x}_n), y_n)}$$

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial W^{(\ell)}} = \frac{1}{N} \sum_{n=1}^N \frac{\partial \mathbf{e}_n}{\partial W^{(\ell)}}$$

We need

$$\frac{\partial \mathbf{e}(\mathbf{x})}{\partial W^{(\ell)}}$$

Numerical Approach

$$\frac{\partial e(\mathbf{x})}{\partial W_{ij}^{(\ell)}} \approx \frac{e(\mathbf{x}|W_{ij}^{(\ell)} + \Delta) - e(\mathbf{x}|W_{ij}^{(\ell)} - \Delta)}{2\Delta}$$

approximate
inefficient



Algorithmic Approach

$e(\mathbf{x})$ is a function of $\mathbf{s}^{(\ell)}$ and $\mathbf{s}^{(\ell)} = (\mathbf{W}^{(\ell)})^T \mathbf{x}^{(\ell-1)}$

$$\begin{aligned} \frac{\partial e}{\partial \mathbf{W}^{(\ell)}} &= \frac{\partial \mathbf{s}^{(\ell)}}{\partial \mathbf{W}^{(\ell)}} \cdot \left(\frac{\partial e}{\partial \mathbf{s}^{(\ell)}} \right)^T && \text{(chain rule)} \\ &= \mathbf{x}^{(\ell-1)} (\boldsymbol{\delta}^{(\ell)})^T \end{aligned}$$

sensitivity

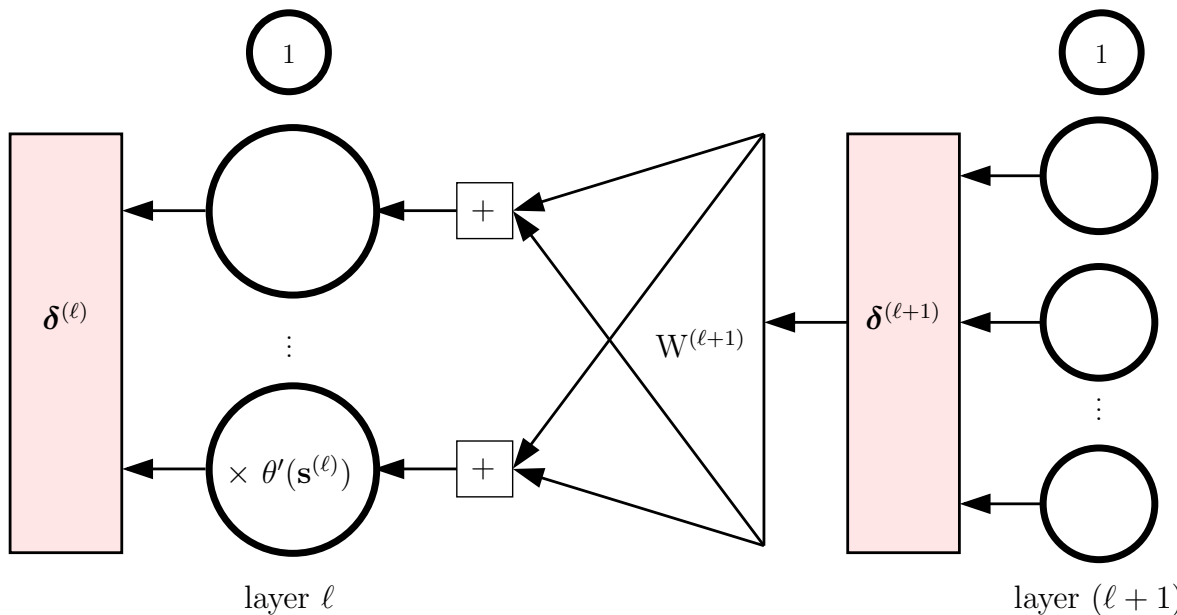
$$\boldsymbol{\delta}^{(\ell)} = \frac{\partial e}{\partial \mathbf{s}^{(\ell)}}$$

Computing $\delta^{(\ell)}$ Using the Chain Rule

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

Multiple applications of the chain rule:

$$\Delta \mathbf{s}^{(\ell)} \xrightarrow{\theta} \Delta \mathbf{x}^{(\ell)} \xrightarrow{W^{(\ell+1)}} \Delta \mathbf{s}^{(\ell+1)} \dots \longrightarrow \Delta \mathbf{e}(\mathbf{x})$$



don't use 0th component (bias)
↓

$$\delta^{(\ell)} = \theta'(\mathbf{s}^{(\ell)}) \otimes [W^{(\ell+1)} \delta^{(\ell+1)}]_1^{d^{(\ell)}}$$

↑
componentwise multiplication

The Backpropagation Algorithm

$$\delta^{(1)} \longleftarrow \delta^{(2)} \dots \longleftarrow \delta^{(L-1)} \longleftarrow \delta^{(L)}$$

Backpropagation to compute sensitivities $\delta^{(\ell)}$:
(Assume $\mathbf{s}^{(\ell)}$ and $\mathbf{x}^{(\ell)}$ have been computed for all ℓ)

- 1: $\delta^{(L)} \longleftarrow 2(x^{(L)} - y) \cdot \theta'(s^{(L)})$ **[Initialization]**
- 2: **for** $\ell = L - 1$ to 1 **do** **[Back-Propagation]**
- 3: Compute (for tanh hidden node):
$$\theta'(\mathbf{s}^{(\ell)}) = \left[1 - \mathbf{x}^{(\ell)} \otimes \mathbf{x}^{(\ell)} \right]_1^{d^{(\ell)}}$$
- 4: $\delta^{(\ell)} \longleftarrow \theta'(\mathbf{s}^{(\ell)}) \otimes \left[\mathbf{W}^{(\ell+1)} \delta^{(\ell+1)} \right]_1^{d^{(\ell)}}$ \longleftarrow componentwise multiplication
- 5: **end for**

Algorithm for Gradient Descent on E_{in}

Algorithm to Compute $E_{\text{in}}(\mathbf{w})$ and $\mathbf{g} = \nabla E_{\text{in}}(\mathbf{w})$:

Input: weights $\mathbf{w} = \{W^{(1)}, \dots, W^{(L)}\}$; data \mathcal{D} .

Output: error $E_{\text{in}}(\mathbf{w})$ and gradient $\mathbf{g} = \{G^{(1)}, \dots, G^{(L)}\}$.

```
1: Initialize:  $E_{\text{in}} = 0$ ; for  $\ell = 1, \dots, L$ ,  $G^{(\ell)} = 0 \cdot W^{(\ell)}$  .
2: for Each data point  $\mathbf{x}_n$  ( $n = 1, \dots, N$ ) do
3:   Compute  $\mathbf{x}^{(\ell)}$  for  $\ell = 0, \dots, L$ . [forward propagation]
4:   Compute  $\boldsymbol{\delta}^{(\ell)}$  for  $\ell = 1, \dots, L$ . [backpropagation]
5:    $E_{\text{in}} \leftarrow E_{\text{in}} + \frac{1}{N}(\mathbf{x}_1^{(L)} - y_n)^2$ .
6:   for  $\ell = 1, \dots, L$  do
7:      $G^{(\ell)}(\mathbf{x}_n) = [\mathbf{x}^{(\ell-1)}(\boldsymbol{\delta}^{(\ell)})^T]$ 
8:      $G^{(\ell)} \leftarrow G^{(\ell)} + \frac{1}{N}G^{(\ell)}(\mathbf{x}_n)$ .
9:   end for
10: end for
```

Can do batch version or sequential version (SGD).

Digits Data

