Learning From Data
Lecture 21
Neural Networks: Backpropagation

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

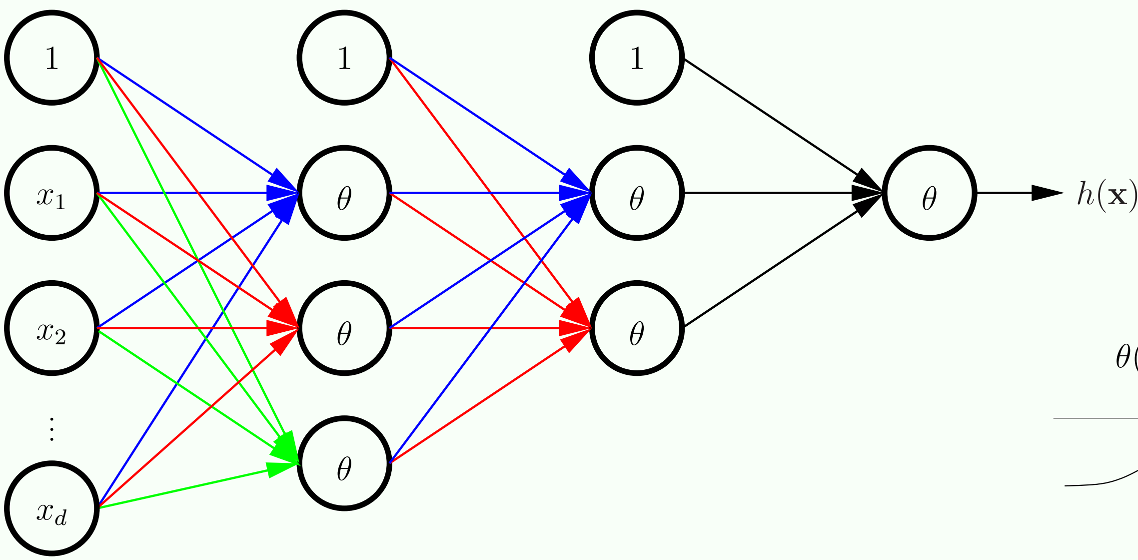
Backpropagation: algorithmic computation of $\dfrac{\partial \mathbf{e}(\mathbf{x})}{\partial \textbf{weights}}$

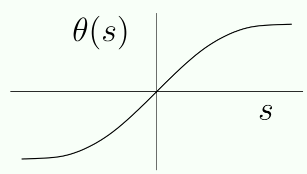
M. Magdon-Ismail
CSCI 4100/6100

RECAP: The Neural Network

Biology

Engineering



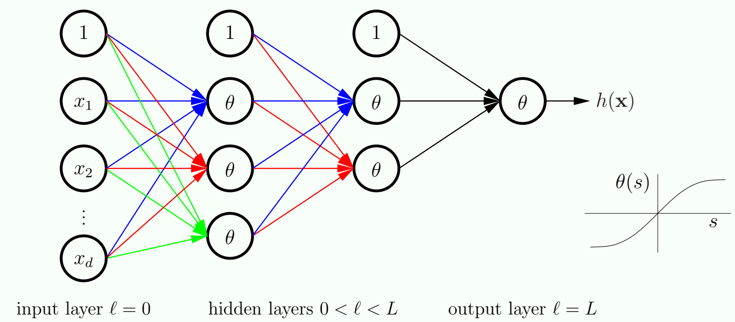


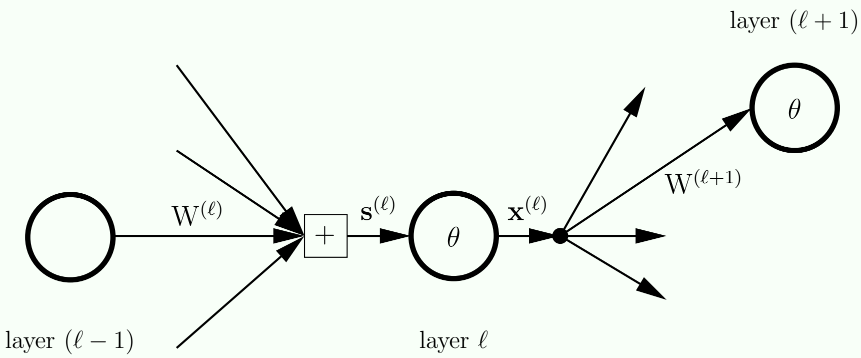
input layer $\ell = 0$

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	$\mathrm{W}^{(\ell)}$	$(d^{(\ell-1)} + 1) \times d^{(\ell)}$ dimensional matrix
weights out	$\mathrm{W}^{(\ell+1)}$	$(d^{(\ell)} + 1) \times d^{(\ell+1)}$ dimensional matrix

layers $\ell = 0, 1, 2, \ldots, L$

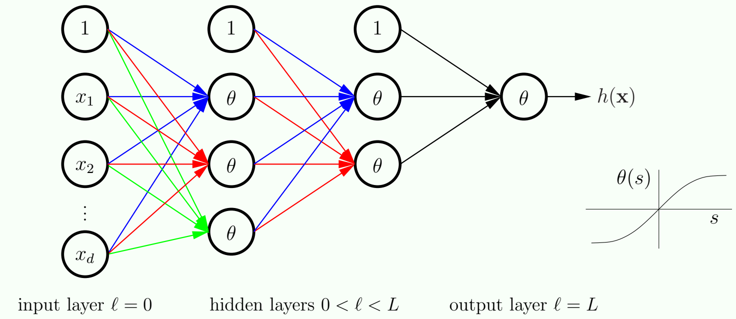
layer ℓ has "dimension" $d^{(\ell)} \implies d^{(\ell)} + 1$ nodes

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

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

The Linear Signal

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



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

$$\begin{bmatrix} s_1^{(\ell)} \\ s_2^{(\ell)} \\ \vdots \\ s_j^{(\ell)} \\ \vdots \\ s_{d^{(\ell)}}^{(\ell)} \end{bmatrix} = \begin{bmatrix} (\mathbf{w}_1^{(\ell)})^{\mathrm{T}} & \rule{2cm}{0.4pt} \\ (\mathbf{w}_2^{(\ell)})^{\mathrm{T}} & \rule{2cm}{0.4pt} \\ & \vdots \\ (\mathbf{w}_j^{(\ell)})^{\mathrm{T}} & \rule{2cm}{0.8pt} \\ & \vdots \\ (\mathbf{w}_{d^{(\ell)}}^{(\ell)})^{\mathrm{T}} & \rule{2cm}{0.4pt} \end{bmatrix} \mathbf{x}^{(\ell-1)}$$

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

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

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

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

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{\mathrm{W}^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{\mathrm{W}^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \cdots \xrightarrow{\mathrm{W}^{(L)}} \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x}).$$

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: $\quad \mathbf{s}^{(\ell)} \leftarrow (\mathrm{W}^{(\ell)})^{\mathrm{T}} \mathbf{x}^{(\ell-1)}$

4: $\quad \mathbf{x}^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(\mathbf{s}^{(\ell)}) \end{bmatrix}$

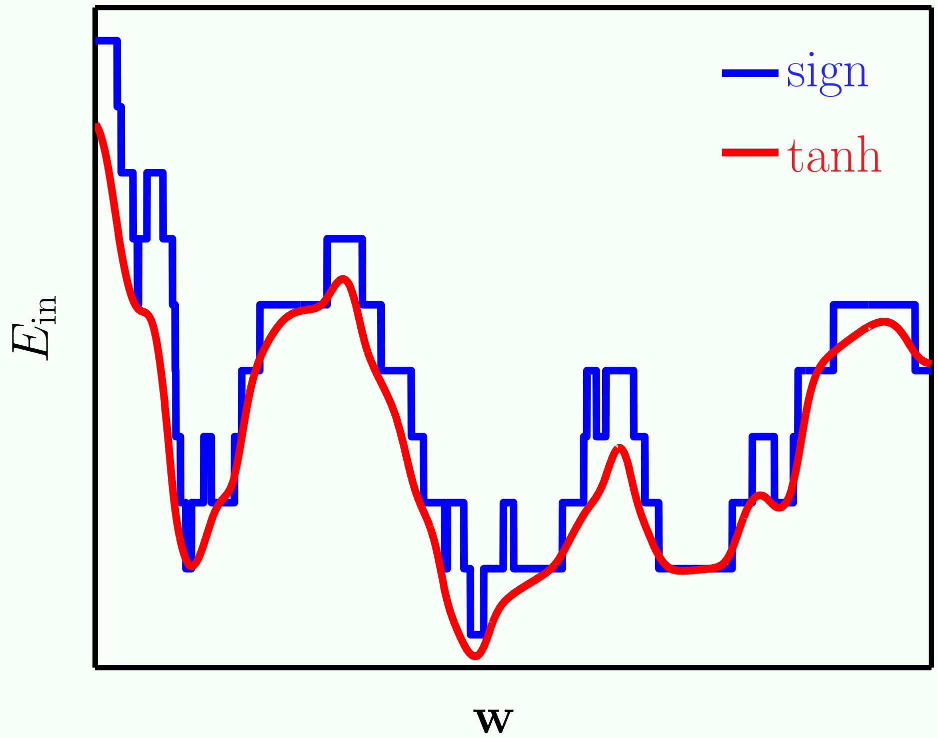
5: **end for**

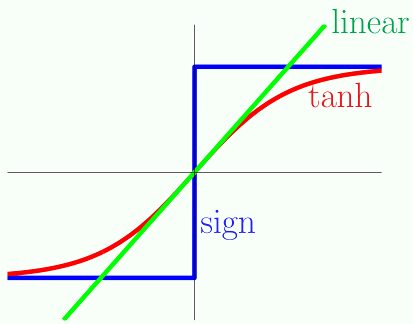
6: $h(\mathbf{x}) = \mathbf{x}^{(L)}$ **[Output]**

Minimizing E_{in}

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

$\mathrm{W} = \{\mathrm{W}^{(1)}, \mathrm{W}^{(2)}, \ldots, \mathrm{W}^{(L)}\}$





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

Gradient Descent

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

Gradient of E_{in}

e_n

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

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

We need

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

Numerical Approach

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

approximate

inefficient

Algorithmic approach ⟶

Algorithmic Approach ☺

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

$$\frac{\partial \mathsf{e}}{\partial \mathrm{W}^{(\ell)}} = \frac{\partial \mathbf{s}^{(\ell)}}{\partial \mathrm{W}^{(\ell)}} \cdot \left(\frac{\partial \mathsf{e}}{\partial \mathbf{s}^{(\ell)}}\right)^{\mathrm{T}} \qquad \text{(chain rule)}$$

$$= \mathbf{x}^{(\ell-1)}(\boldsymbol{\delta}^{(\ell)})^{\mathrm{T}}$$

sensitivity

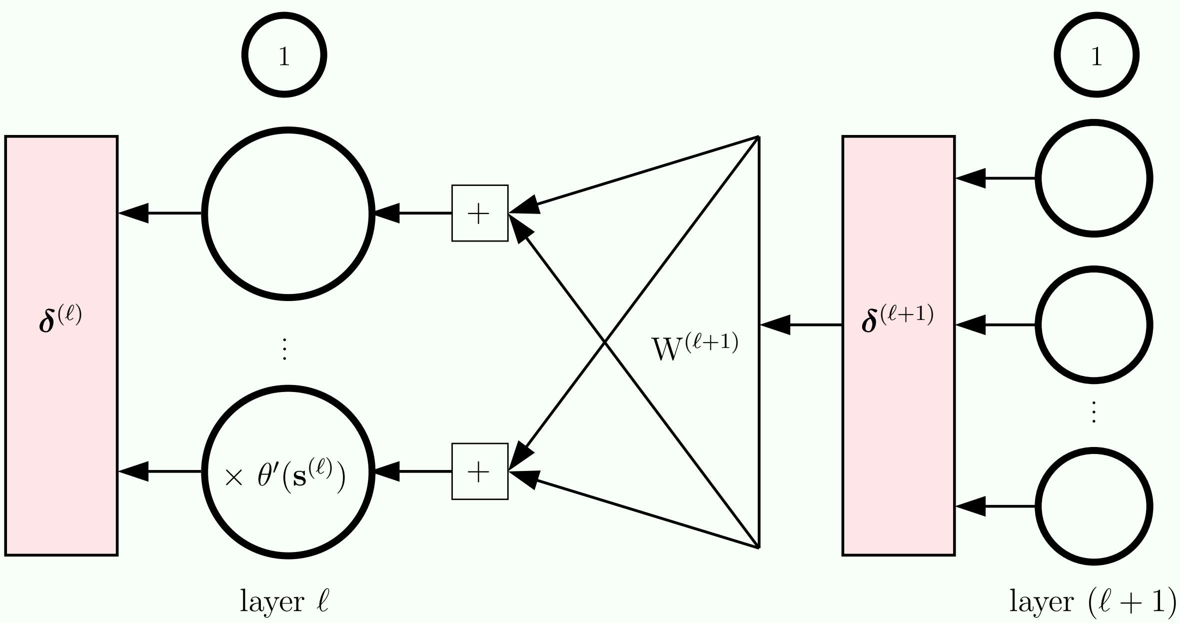
$$\boldsymbol{\delta}^{(\ell)} = \frac{\partial \mathsf{e}}{\partial \mathbf{s}^{(\ell)}}$$

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

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

Multiple applications of the chain rule:

$$\color{red}{\Delta\mathbf{s}^{(\ell)}} \xrightarrow{\theta} \Delta\mathbf{x}^{(\ell)} \xrightarrow{\mathrm{W}^{(\ell+1)}} \color{red}{\Delta\mathbf{s}^{(\ell+1)}} \cdots \longrightarrow \Delta\mathrm{e}(\mathbf{x})$$



don't use 0^{th} component (bias)

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

componentwise multiplication

The Backpropagation Algorithm

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

Backpropagation to compute sensitivities $\boldsymbol{\delta}^{(\ell)}$:

(Assume $\mathbf{s}^{(\ell)}$ and $\mathbf{x}^{(\ell)}$ have been computed for all ℓ)

$\delta^{(L)} \leftarrow 2(x^{(L)} - y) \cdot \theta'(s^{(L)})$ **[Initialization]**

for $\ell = L-1$ to 1 **do** **[Back-Propagation]**

Compute (for tanh hidden node):

$$\theta'(\mathbf{s}^{(\ell)}) = \left[1 - \mathbf{x}^{(\ell)} \otimes \mathbf{x}^{(\ell)}\right]_1^{d^{(\ell)}}$$

$$\boldsymbol{\delta}^{(\ell)} \leftarrow \theta'(\mathbf{s}^{(\ell)}) \otimes \left[\mathrm{W}^{(\ell+1)}\boldsymbol{\delta}^{(\ell+1)}\right]_1^{d^{(\ell)}}$$

$\longleftarrow$ componentwise multiplication

end for

Gradient Descent on E_{in} $\longrightarrow$

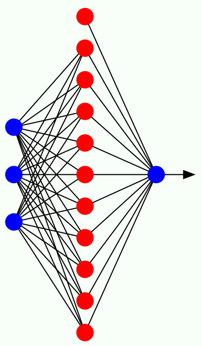
Algorithm for Gradient Descent on E_{in}

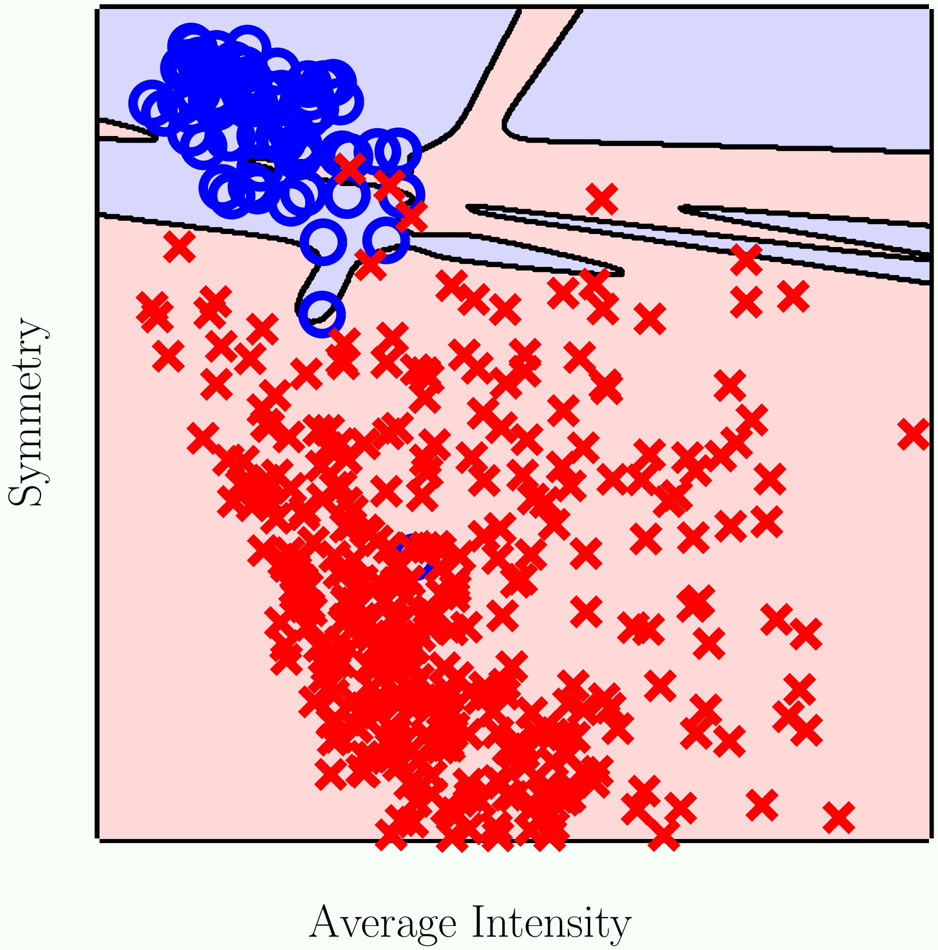
Algorithm to Compute $E_{\text{in}}(\mathbf{w})$ and $\mathbf{g} = \nabla E_{\text{in}}(\mathbf{w})$:
Input: weights $\mathbf{w} = \{\mathrm{W}^{(1)}, \ldots, \mathrm{W}^{(L)}\}$; data $\mathcal{D}$.
Output: error $E_{\text{in}}(\mathbf{w})$ and gradient $\mathbf{g} = \{\mathrm{G}^{(1)}, \ldots, \mathrm{G}^{(L)}\}$.

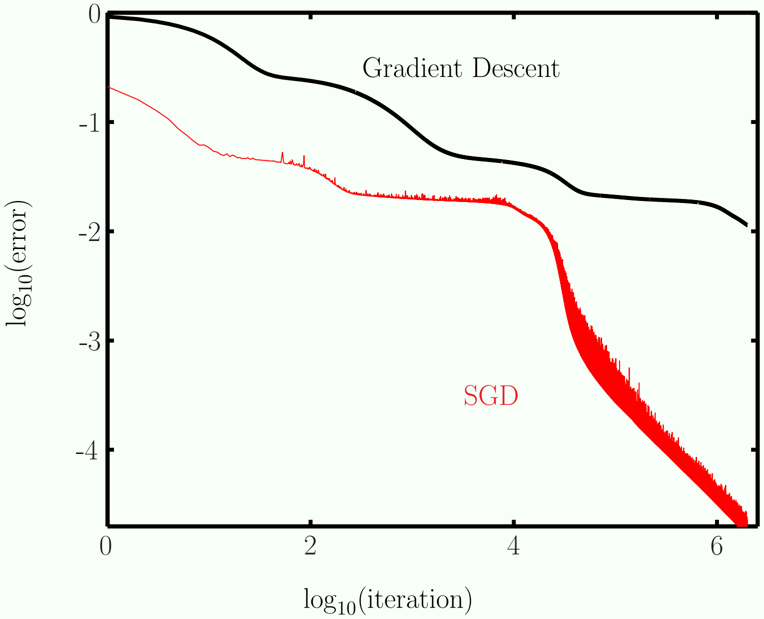
Initialize: $E_{\text{in}} = 0$; for $\ell = 1, \ldots, L$, $\mathrm{G}^{(\ell)} = 0 \cdot \mathrm{W}^{(\ell)}$.
for Each data point $\mathbf{x}_n$ $(n = 1, \ldots, N)$ **do**
  Compute $\mathbf{x}^{(\ell)}$ for $\ell = 0, \ldots, L$. [forward propagation]
  Compute $\boldsymbol{\delta}^{(\ell)}$ for $\ell = 1, \ldots, L$. [backpropagation]
  $E_{\text{in}} \leftarrow E_{\text{in}} + \frac{1}{N}(\mathbf{x}_1^{(L)} - y_n)^2$.
  for $\ell = 1, \ldots, L$ **do**
    $\mathrm{G}^{(\ell)}(\mathbf{x}_n) = [\mathbf{x}^{(\ell-1)}(\boldsymbol{\delta}^{(\ell)})^{\mathrm{T}}]$
    $\mathrm{G}^{(\ell)} \leftarrow \mathrm{G}^{(\ell)} + \frac{1}{N}\mathrm{G}^{(\ell)}(\mathbf{x}_n)$.
  end for
end for

Can do batch version or sequential version (SGD).

Digits Data




Symmetry
Average Intensity


Gradient Descent
SGD
$\log_{10}(\text{error})$
$\log_{10}(\text{iteration})$
0
-1
-2
-3
-4
0
2
4
6