

Outline

- Stochastic gradient descent
- Neural network model
- Backpropagation algorithm

Applying SGD

All the weights $\mathbf{w} = \{w_{ij}^{(l)}\}$ determine $h(\mathbf{x})$

Error on example (\mathbf{x}_n, y_n) is

$$e(h(\mathbf{x}_n), y_n) = e(\mathbf{w})$$

To implement SGD, we need the gradient

$$\nabla e(\mathbf{w}): \frac{\partial e(\mathbf{w})}{\partial w_{ij}^{(l)}} \text{ for all } i, j, l$$

Computing $\frac{\partial e(\mathbf{w})}{\partial w_{ij}^{(l)}}$

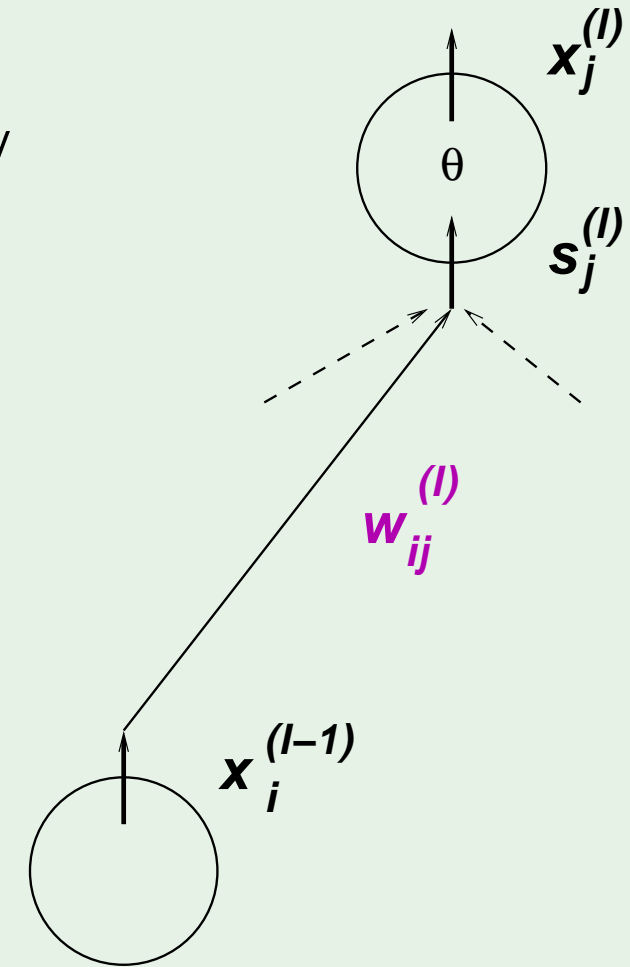
We can evaluate $\frac{\partial e(\mathbf{w})}{\partial w_{ij}^{(l)}}$ one by one: analytically or numerically

A trick for efficient computation:

$$\frac{\partial e(\mathbf{w})}{\partial w_{ij}^{(l)}} = \frac{\partial e(\mathbf{w})}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}$$

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

We only need: $\frac{\partial e(\mathbf{w})}{\partial s_j^{(l)}} = \delta_j^{(l)}$



δ for the final layer

$$\delta_j^{(l)} = \frac{\partial e(\mathbf{w})}{\partial s_j^{(l)}}$$

For the final layer $l = L$ and $j = 1$:

$$\delta_1^{(L)} = \frac{\partial e(\mathbf{w})}{\partial s_1^{(L)}}$$

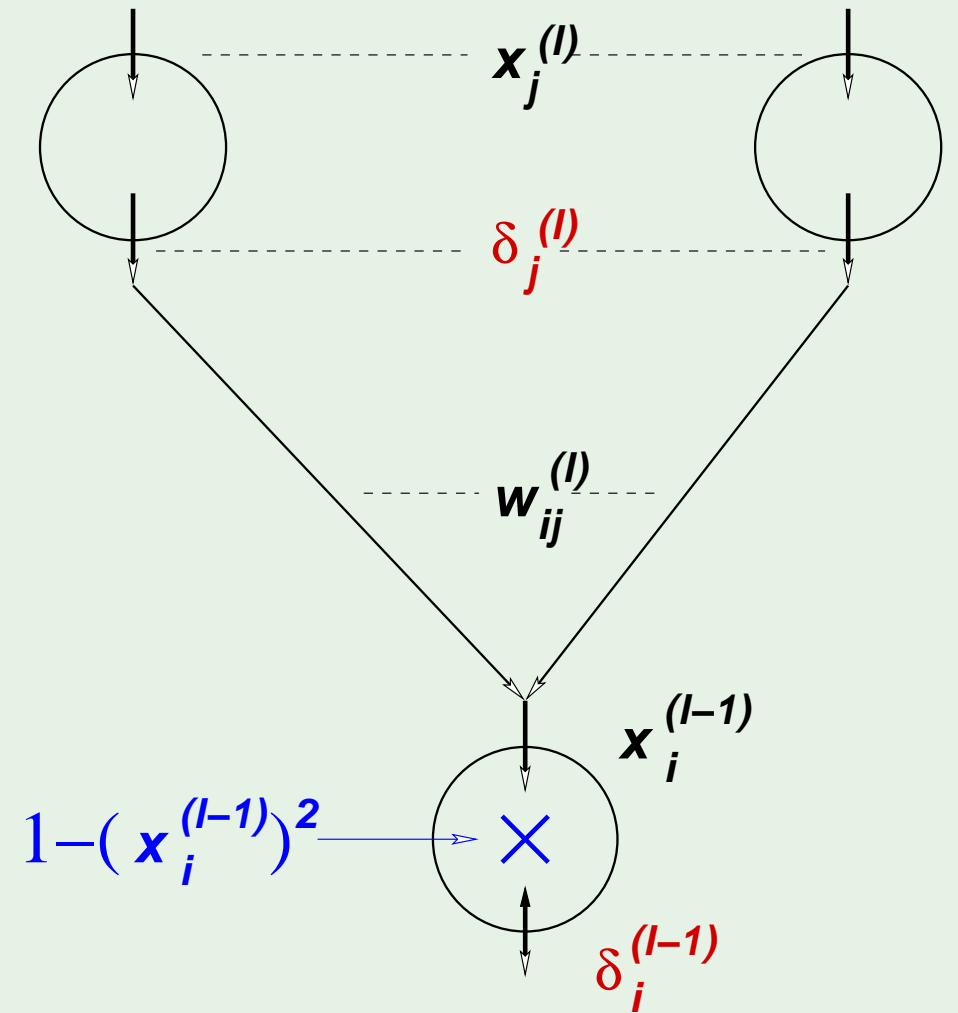
$$e(\mathbf{w}) = (x_1^{(L)} - y_n)^2$$

$$x_1^{(L)} = \theta(s_1^{(L)})$$

$$\theta'(s) = 1 - \theta^2(s) \quad \text{for the tanh}$$

Back propagation of δ

$$\begin{aligned}
 \delta_i^{(l-1)} &= \frac{\partial e(\mathbf{w})}{\partial s_i^{(l-1)}} \\
 &= \sum_{j=1}^{d^{(l)}} \frac{\partial e(\mathbf{w})}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial x_i^{(l-1)}} \times \frac{\partial x_i^{(l-1)}}{\partial s_i^{(l-1)}} \\
 &= \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \times \theta'(s_i^{(l-1)}) \\
 \delta_i^{(l-1)} &= (1 - (x_i^{(l-1)})^2) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)} \delta_j^{(l)}
 \end{aligned}$$



Backpropagation algorithm

- 1: Initialize all weights $w_{ij}^{(l)}$ **at random**
- 2: **for** $t = 0, 1, 2, \dots$ **do**
- 3: Pick $n \in \{1, 2, \dots, N\}$
- 4: *Forward:* Compute all $x_j^{(l)}$
- 5: *Backward:* Compute all $\delta_j^{(l)}$
- 6: Update the weights: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$
- 7: Iterate to the next step until it is time to stop
- 8: Return the final weights $w_{ij}^{(l)}$

