Artificial Intelligence
CS-401

Chapter 18

Artificial Neural Networks

Backpropagation

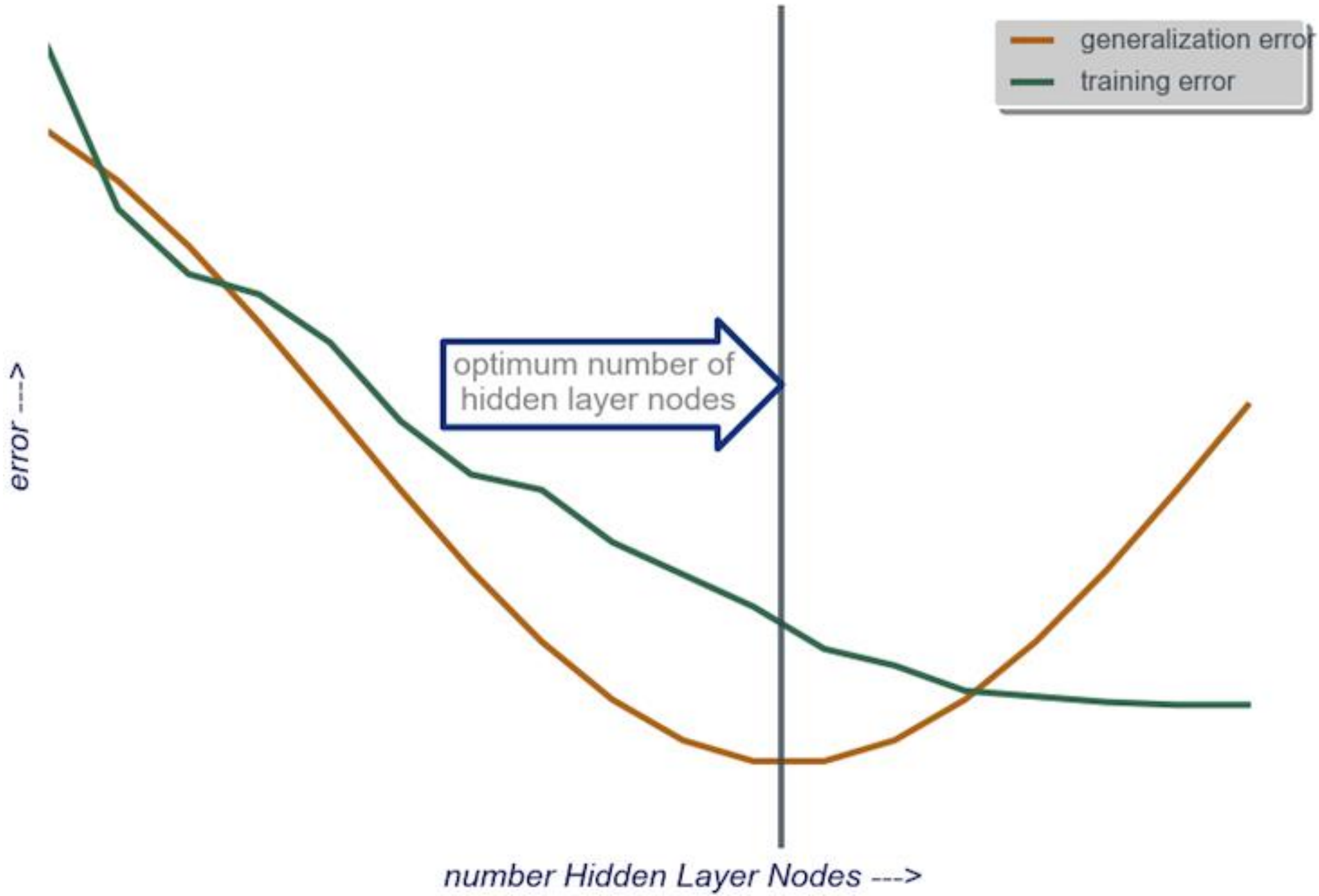
Dr. Hafeez Ur Rehman

(Email: hafeez.urrehman@nu.edu.pk)

We can derive gradient descent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units $\rightarrow$ Backpropagation

Hidden Layer Node Selection



Notations for Multi-Layer ANN

- x_{ji} = the ith input to unit j
- w_{ji} = the weight associated with the ith input to unit j
- $net_j = \sum_i w_{ji} x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output computed by unit j
- t_j = the target output for unit j
- σ = the sigmoid function
- $outputs$ = the set of units in the final layer of the network
- $Downstream(j)$ = the set of units whose immediate inputs include the output of unit j

Error Gradient for a Sigmoid Unit

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \; \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \left(- \frac{\partial o_d}{\partial w_i} \right) \\
&= - \sum_d (t_d - o_d) \; \frac{\partial o_d}{\partial net_d} \; \frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

But we know:

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$

Let: $\delta_k = -\frac{\partial E}{\partial net_k}$

Since for multi-layer network any output layer weight affects only one output layer perceptron therefore this is the same weight update rule for output layer perceptron k.

Error for Hidden Node j

$$
\begin{aligned}
\frac{\partial E}{\partial net_j} &= \sum_{k \in Outs(j)} \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Outs(j)} -\delta_k \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Outs(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \\
&= \sum_{k \in Outs(j)} -\delta_k \, w_{kj} \frac{\partial o_k}{\partial net_j} \\
&= \sum_{k \in Outs(j)} -\delta_k \, w_{kj} \, o_j (1 - o_j) \\
\delta_j &= -\frac{\partial E}{\partial net_j} = o_j (1 - o_j) \sum_{k \in Outs(j)} \delta_k \, w_{kj}
\end{aligned}
$$

Backpropagation Algorithm

Initialize all weights to small random numbers

Until convergence, Do

For each training example, Do

1. Input it to network and compute network outputs
2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where $\Delta w_{i,j} = \eta \delta_j x_{i,j}$