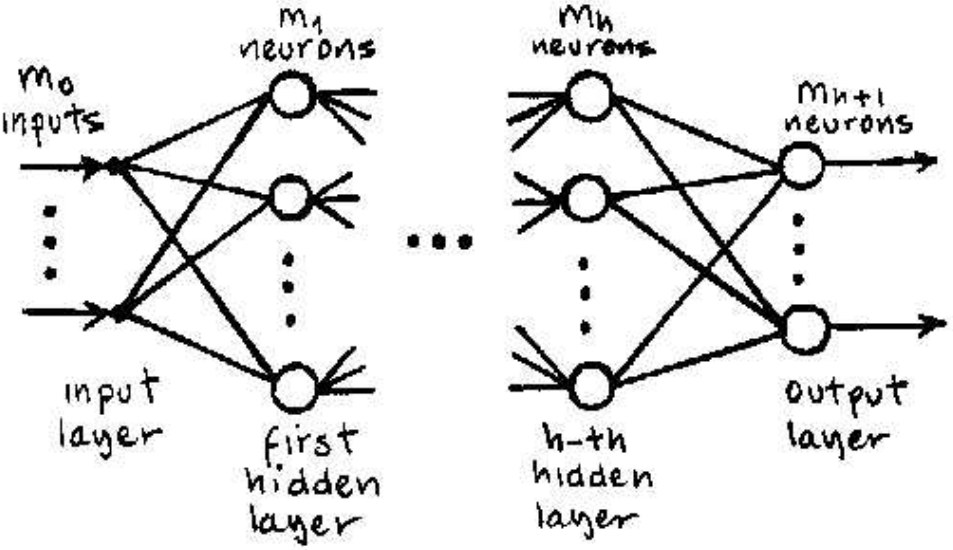
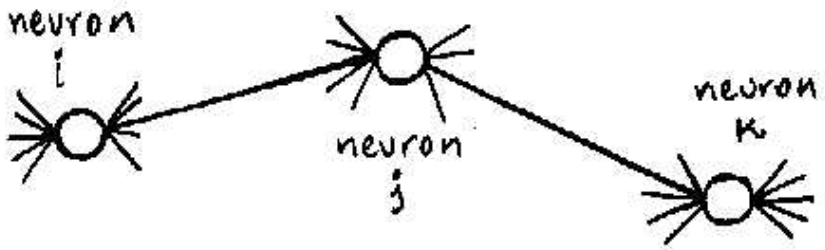
BACK-PROPAGATION

Back-propagation (BP), a landmark in ANN

Multilayer perceptron with h hidden layers:



Indices i, j, k refer to neurons in 3 consecutive layers:



The error signal at the output of neuron j at iteration n (i.e., presentation of the n-th training sample) is

$$e_j(n) = d_j(n) - y_j(n)$$

where $d_j(n)$ is the desired response for neuron j.

Dr. J.E. Rayas-Sánchez

The total error over all neurons in the l-th layer is

$$E(n) = \frac{1}{2} \sum_{j=1}^{m_l} e_j^2(n)$$

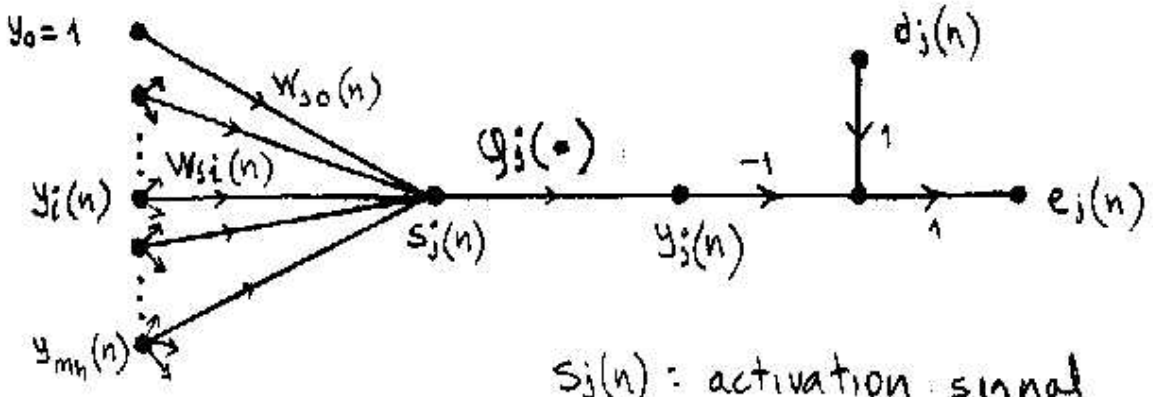
m_l is the number of neurons in the l-th hidden layer.

For a training set with N learning samples, the average square learning error is

$$E_{av} = \frac{1}{N} \sum_{n=1}^{N} E_{out}(n) \quad \text{where} \quad E_{out}(n) = \frac{1}{2} \sum_{j=1}^{m_{n+1}} e_j^2(n)$$

E_{av} is the objective function in the BP algorithm

Signal-flow graph for output neuron j:



$s_j(n)$: activation signal
$w_{ji}(n)$: synaptic weight
φ_j : activation function

$$s_j(n) = \sum_{i=0}^{m_n} w_{ji}(n)\, y_i(n) \quad , \quad y_j(n) = \varphi_j\big(s_j(n)\big)$$

BP is a gradient-based method:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

where η is the learning rate

We can show that

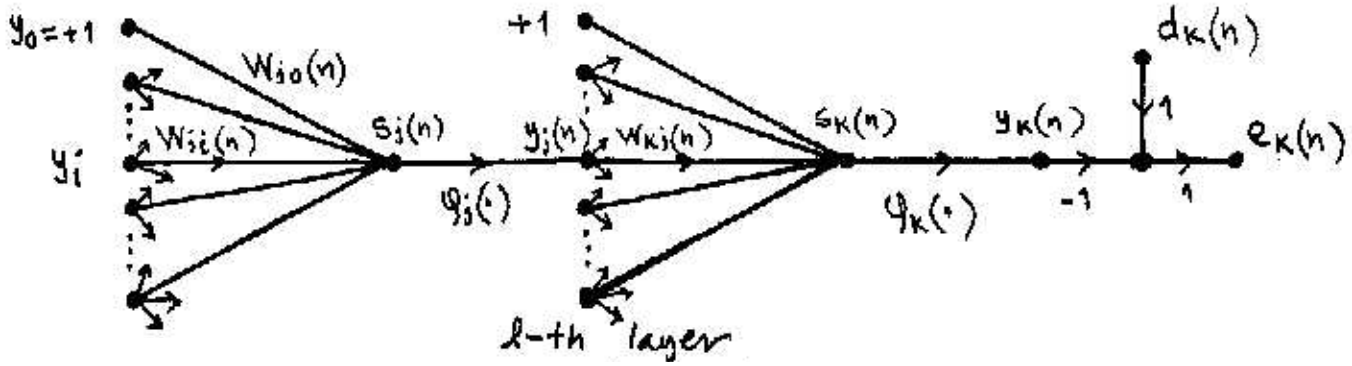
$$\Delta w_{ji}(n) = \eta \, \delta_j(n) \, y_i(n)$$

where

$$\delta_j(n) = e_j(n) \, \varphi_j'\left(s_j(n)\right)$$

↳ local gradient at neuron j

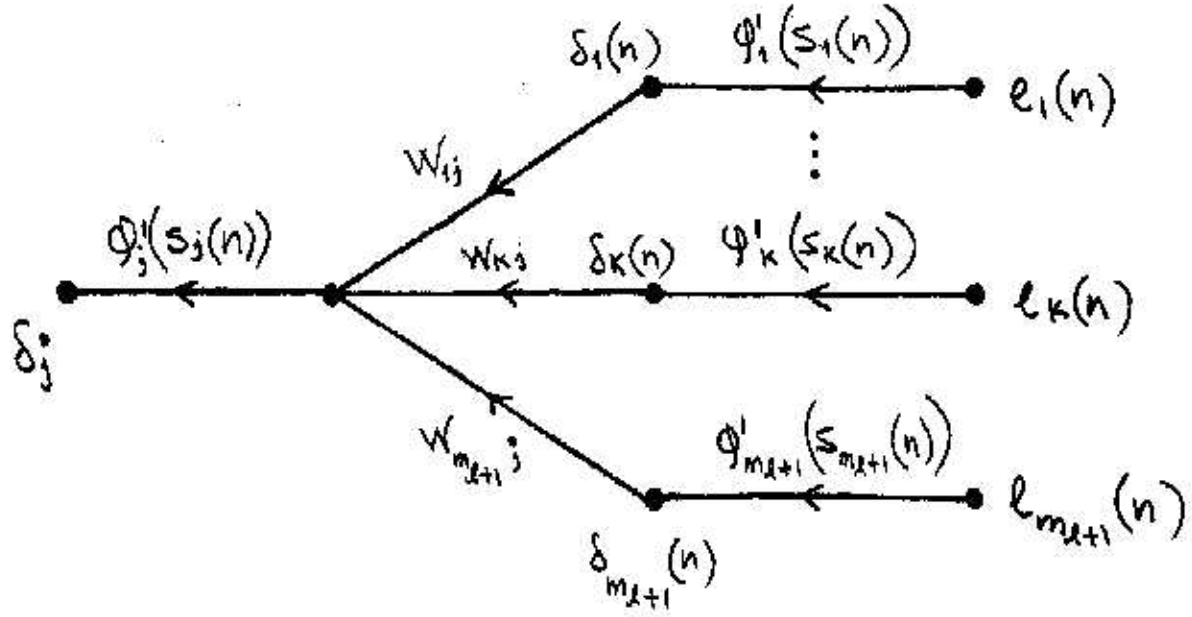
Signal-flow graph for a hidden neuron j



We can show that the local gradient at neuron j is:

$$\delta_j(n) = \varphi_j'(s_j(n)) \sum_{k=1}^{m_{l+1}} \delta_k(n)\, w_{kj}(n)$$

Back-propagation formula for the local gradient



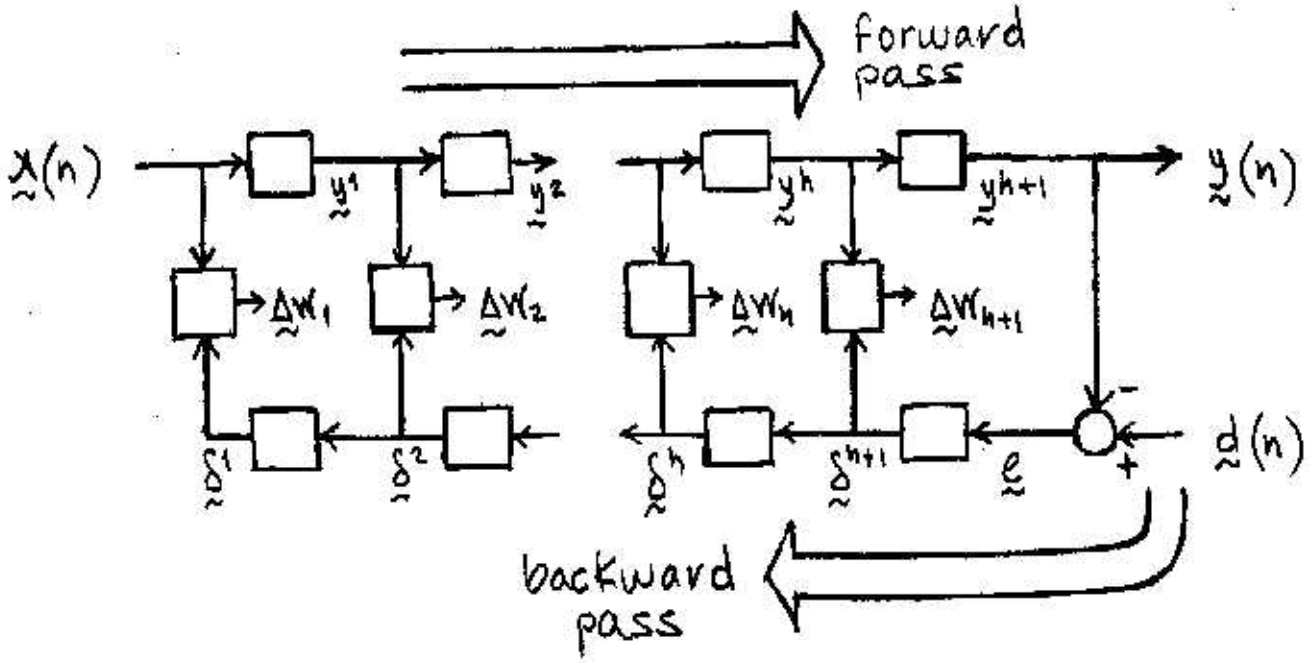
In summary, the BP updating formula for the synaptic weights in the l-th layer is

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, y_i(n) \qquad i = 0, 1, \ldots m_{l-1}, \quad j = 0, 1, \ldots m_l$$

η: learning rate; $\delta_j(n)$: local gradient; $y_i(n)$: output of neuron i

$$\delta_j(n) = \begin{cases} \varphi_j'(s_j(n))\, e_j(n) & \text{if neuron } j \text{ is in the output layer} \\ \varphi_j'(s_j(n)) \sum_{k=1}^{m_{l+1}} \delta_k(n) w_{kj}(n) & \text{if neuron } j \text{ is in a hidden layer} \end{cases}$$

The BP algorithm is executed in 2 passes:



* Effects of h (number of hidden layers)

Increasing h...

i) increases the computational effort per iteration in the BP algorithm.

ii) increases the ability of the ANN to approximate complex input-output mappings.

iii) increases the number of learning samples (N) needed to achieve a generalization error ϵ (Haykin, 1999)

$$N = O\left(\frac{W}{\epsilon}\right)$$

W: number of free parameters in the ANN

$\Rightarrow$

if N is small, use the simplest ANN

iv) allows more effective representation of hierarchical information in the original problem. (Gupta and Zhang, 1999).

BP is a gradient-based method:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

η: learning rate

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial s_j(n)} \frac{\partial s_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\partial}{\partial e_j(n)} \left[\frac{1}{2} \sum_{j=1}^{m_2} e_j^2(n) \right] = e_j(n)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)} \left[d_j(n) - y_j(n) \right] = -1$$

$$\frac{\partial y_j(n)}{\partial s_j(n)} = \frac{\partial}{\partial s_j(n)} \left[\varphi_j(s_j(n)) \right] = \varphi_j'(s_j(n))$$

$$\frac{\partial s_j(n)}{\partial w_{ji}(n)} = \frac{\partial}{\partial w_{ji}(n)} \left[\sum_{i=0}^{m_1} w_{ji}(n) y_i(n) \right] = y_i(n)$$

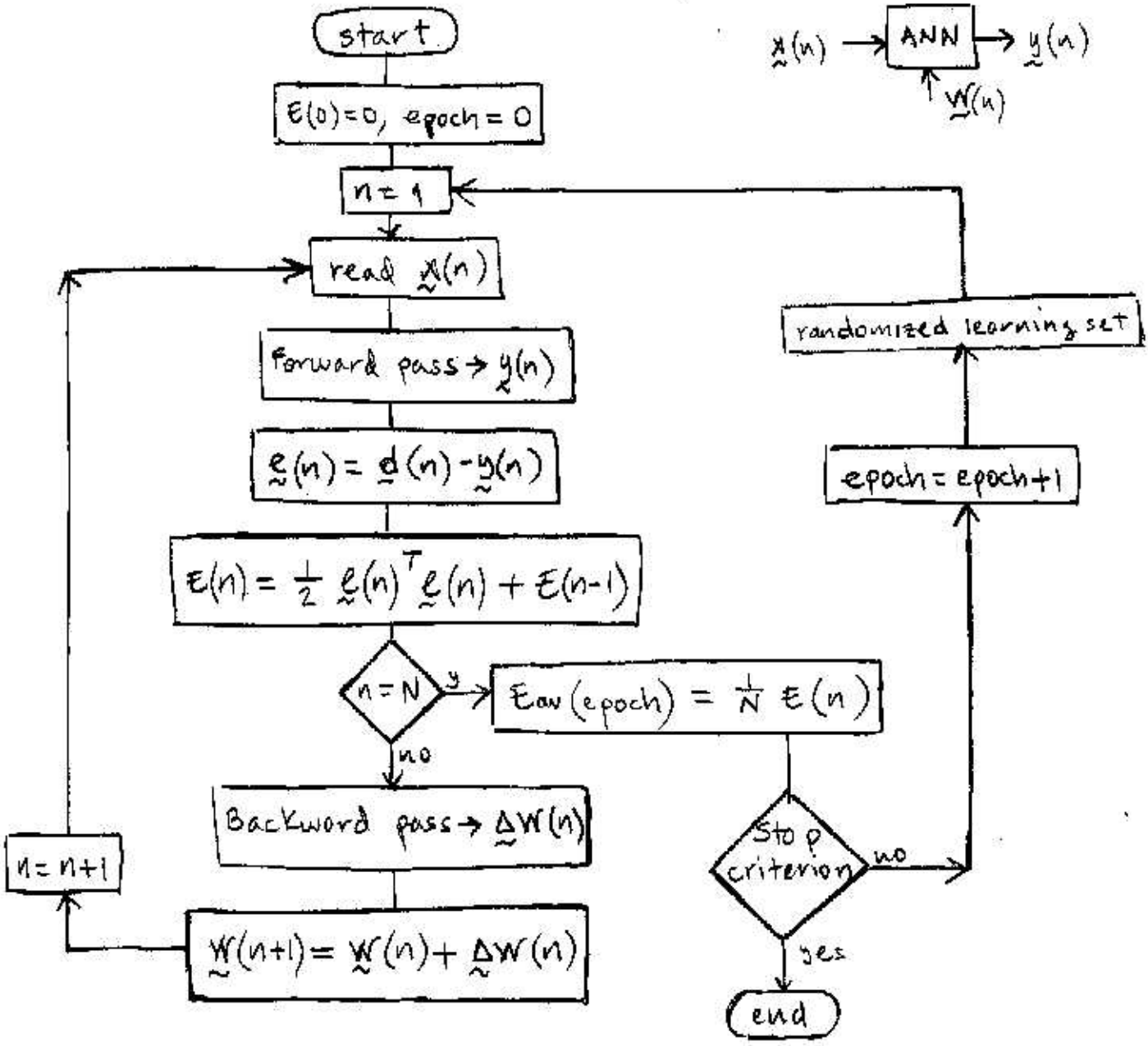
Defining the local gradient at neuron j (δ_j) as

$$\delta_j(n) = -\frac{\partial E(n)}{\partial s_j(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial s_j(n)} = e_j(n) \varphi_j'(s_j(n))$$

then

$$\Delta w_{ji}(n) = \eta \, \delta_j(n) y_i(n)$$

BP ALGORITHM IN SEQUENTIAL MODE



BP ALGORITHM IN BATCH MODE.