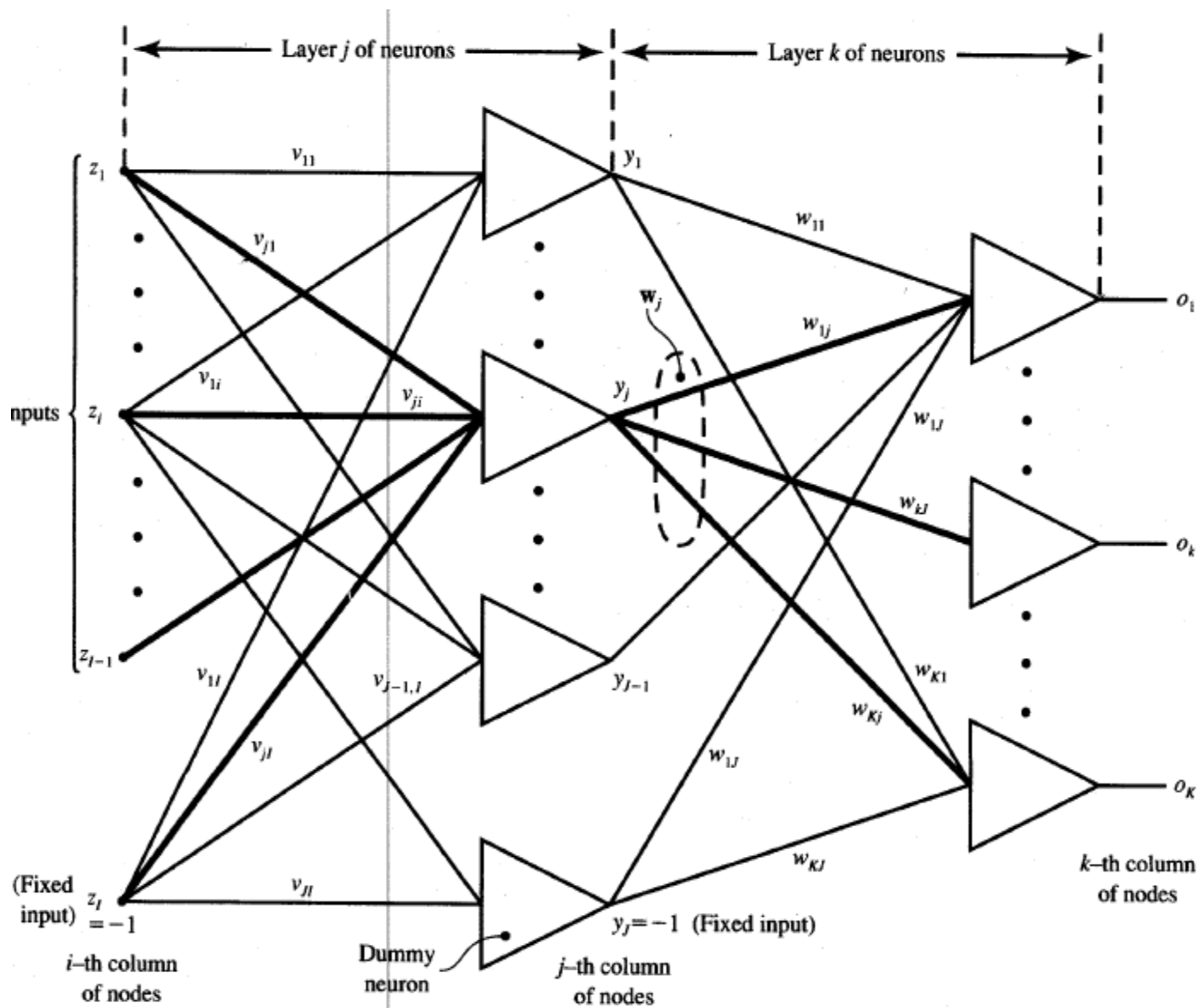


# **ERROR BACKPROPAGATION TRAINING**



Layered feed forward network with hidden and output layers

# EBPTA

Given are  $P$  training pairs

$$\{\mathbf{z}_1, \mathbf{d}_1, \mathbf{z}_2, \mathbf{d}_2, \dots, \mathbf{z}_P, \mathbf{d}_P\},$$

where  $\mathbf{z}_i$  is  $(I \times 1)$ ,  $\mathbf{d}_i$  is  $(K \times 1)$ , and  $i = 1, 2, \dots, P$ . Note that the  $I$ 'th component of each  $\mathbf{z}_i$  is of value  $-1$  since input vectors have been augmented. Size  $J - 1$  of the hidden layer having outputs  $\mathbf{y}$  is

selected. Note that the  $J$ 'th component of  $\mathbf{y}$  is of value  $-1$ , since hidden layer outputs have also been augmented;  $\mathbf{y}$  is  $(J \times 1)$  and  $\mathbf{o}$  is  $(K \times 1)$ .

# EBPTA

Step 1:  $\eta > 0$ ,  $E_{\max}$  chosen.

Weights  $\mathbf{W}$  and  $\mathbf{V}$  are initialized at small random values;  $\mathbf{W}$  is  $(K \times J)$ ,  $\mathbf{V}$  is  $(J \times I)$ .

$$q \leftarrow 1, p \leftarrow 1, E \leftarrow 0$$

Step 2: Training step starts here (See Note 1 at end of list.)

Input is presented and the layers' outputs computed [ $f(\text{net})$  as in (2.3a) is used]:

$$\mathbf{z} \leftarrow \mathbf{z}_p, \mathbf{d} \leftarrow \mathbf{d}_p$$

$$y_j \leftarrow f(\mathbf{v}_j^t \mathbf{z}), \quad \text{for } j = 1, 2, \dots, J$$

where  $\mathbf{v}_j$ , a column vector, is the  $j$ 'th row of  $\mathbf{V}$ , and

$$o_k \leftarrow f(\mathbf{w}_k^t \mathbf{y}), \quad \text{for } k = 1, 2, \dots, K$$

where  $\mathbf{w}_k$ , a column vector, is the  $k$ 'th row of  $\mathbf{W}$ .

Step 3: Error value is computed:

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E, \quad \text{for } k = 1, 2, \dots, K$$

**Step 4:** Error signal vectors  $\delta_o$  and  $\delta_y$  of both layers are computed.

Vector  $\delta_o$  is  $(K \times 1)$ ,  $\delta_y$  is  $(J \times 1)$ . (See Note 2 at end of list.)

The error signal terms of the output layer in this step are

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \quad \text{for } k = 1, 2, \dots, K$$

The error signal terms of the hidden layer in this step are

$$\delta_{yj} = \frac{1}{2}(1 - y_j^2) \sum_{k=1}^K \delta_{ok} w_{kj}, \quad \text{for } j = 1, 2, \dots, J$$

**Step 5:** Output layer weights are adjusted:

$$w_{kj} \leftarrow w_{kj} + \eta \delta_{ok} y_j, \quad \text{for } k = 1, 2, \dots, K \text{ and} \\ j = 1, 2, \dots, J$$

**Step 6:** Hidden layer weights are adjusted:

$$v_{ji} \leftarrow v_{ji} + \eta \delta_{yj} z_i, \quad \text{for } j = 1, 2, \dots, J \text{ and} \\ i = 1, 2, \dots, I$$

**Step 7:** If  $p < P$  then  $p \leftarrow p + 1$ ,  $q \leftarrow q + 1$ , and go to Step 2;  
otherwise, go to Step 8.

# EBPTA

Step 8: The training cycle is completed.

For  $E < E_{\max}$  terminate the training session. Output weights  $W$ ,  $V$ ,  $q$ , and  $E$ .

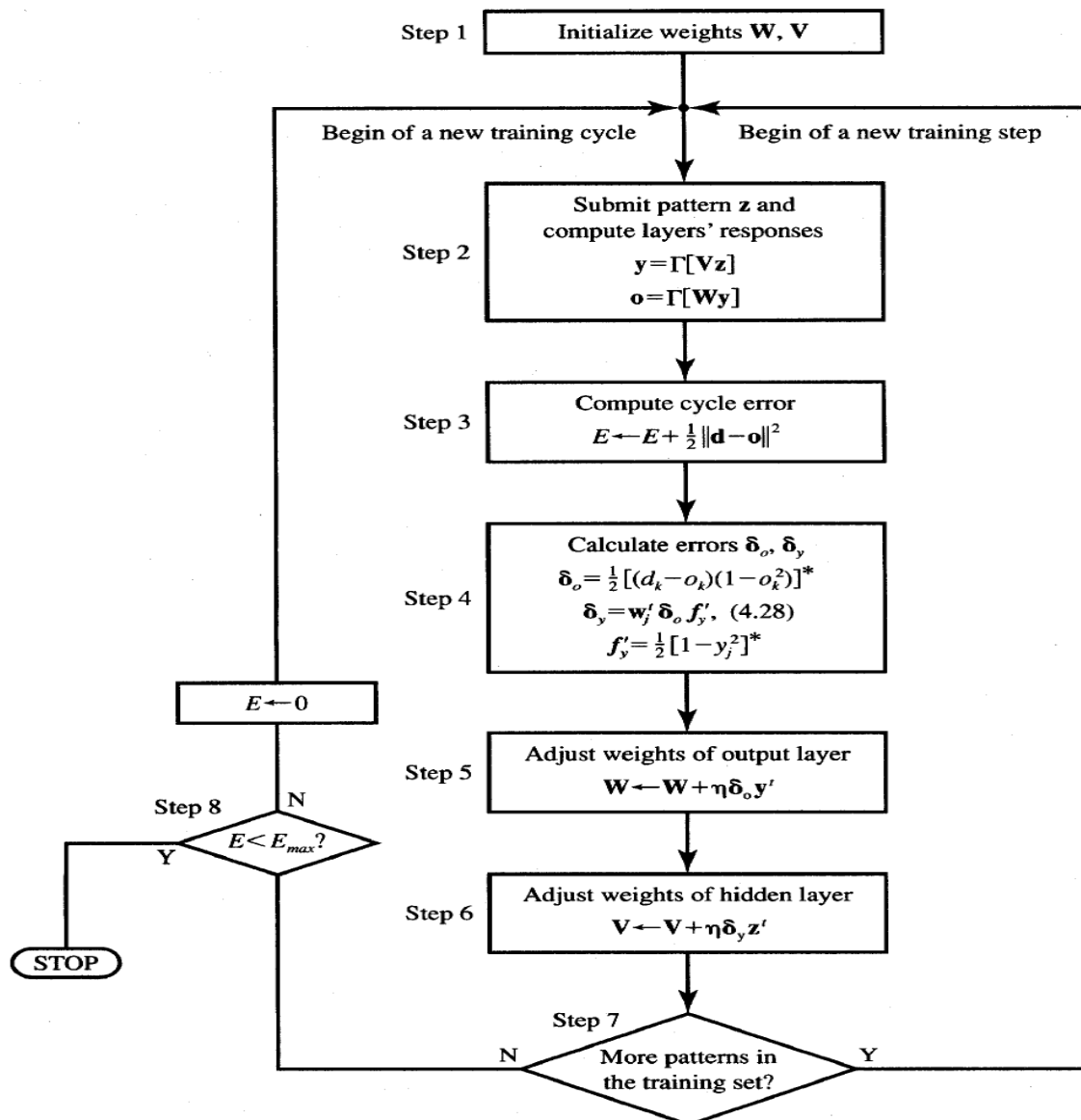
If  $E > E_{\max}$ , then  $E \leftarrow 0$ ,  $p \leftarrow 1$ , and initiate the new training cycle by going to Step 2.

■ NOTE 1 For best results, patterns should be chosen at random from the training set (justification follows in Section 4.5).

■ NOTE 2 If formula (2.4a) is used in Step 2, then the error signal terms in Step 4 are computed as follows

$$\delta_{ok} = (d_k - o_k)(1 - o_k)o_k, \quad \text{for } k = 1, 2, \dots, K$$

$$\delta_{yj} = y_j(1 - y_j) \sum_{k=1}^K \delta_{ok} w_{kj}, \quad \text{for } j = 1, 2, \dots, J$$

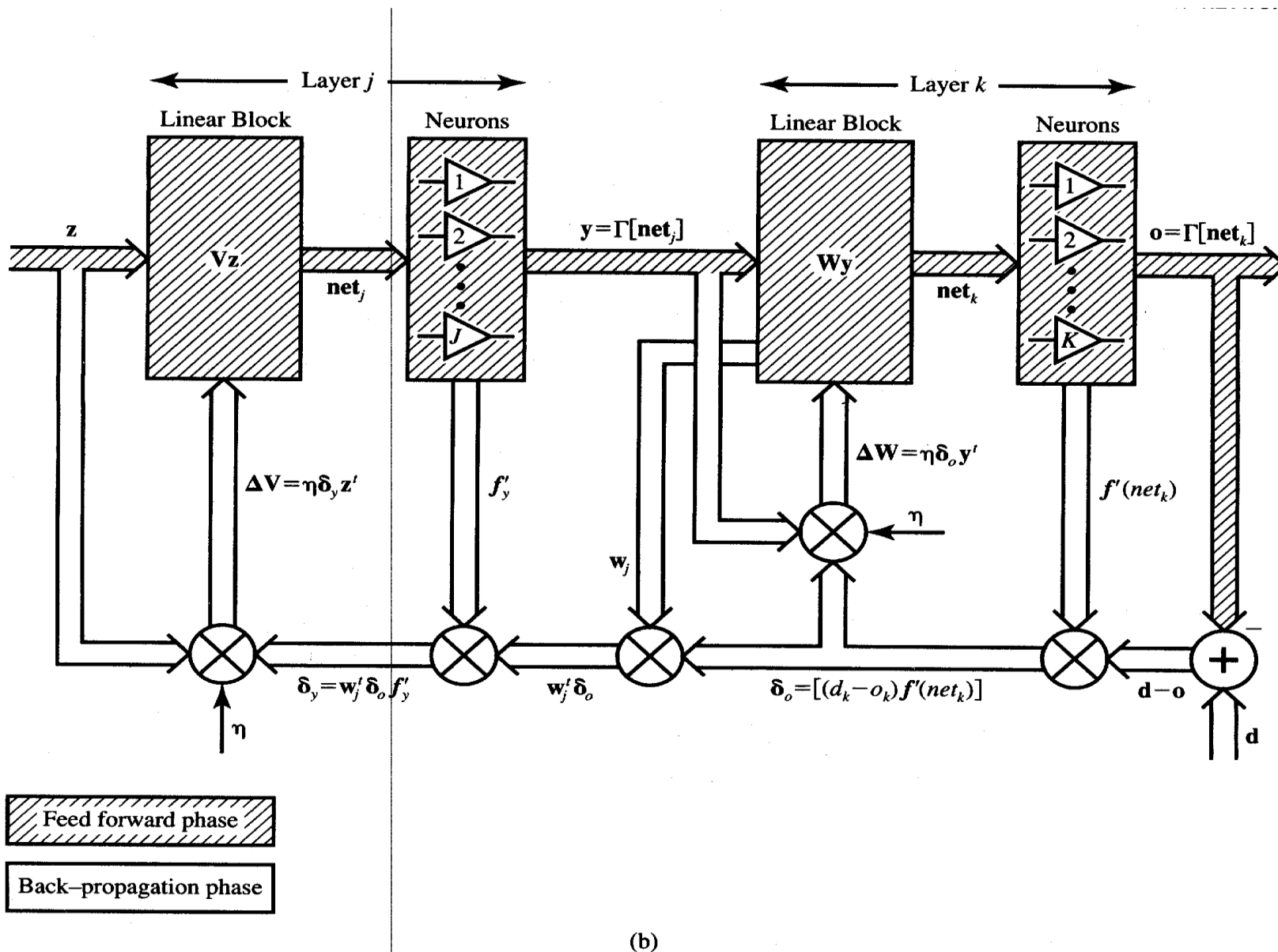


\*If  $f(net)$  given by (2.4a) is used in Step 2, then in Step 4 use

$$\delta_o = [(d_k - o_k)(1 - o_k)o_k], \quad f'_y = [(1 - y_j)y_j]$$

(a)

**Figure 4.8a** Error back-propagation training (EBPT algorithm): (a) algorithm flowchart.



**Figure 4.8b** Error back-propagation training (EBPT algorithm) (*continued*): (b) block diagram illustrating forward and backward signal flow.



# Learning Factors

# Initial Weights

- Initialization strongly affects the ultimate solution
- Initial weights with equal weight values → if the solution requires unequal weights to be developed → wrong training
- Network disturbance by random factors or the random character of input patterns during training → result in symmetric weights
- Continuing training beyond a certain low-level results in the undesirable drift of weights-
  - network learning should be restarted with other random weights

# Cumulative weight Adjustment versus incremental Updating

- Cumulative weight adjustment after each completed training cycle
- Advisable to use the incremental weight updating after each pattern presentation, but choose patterns in a random sequence from a training set

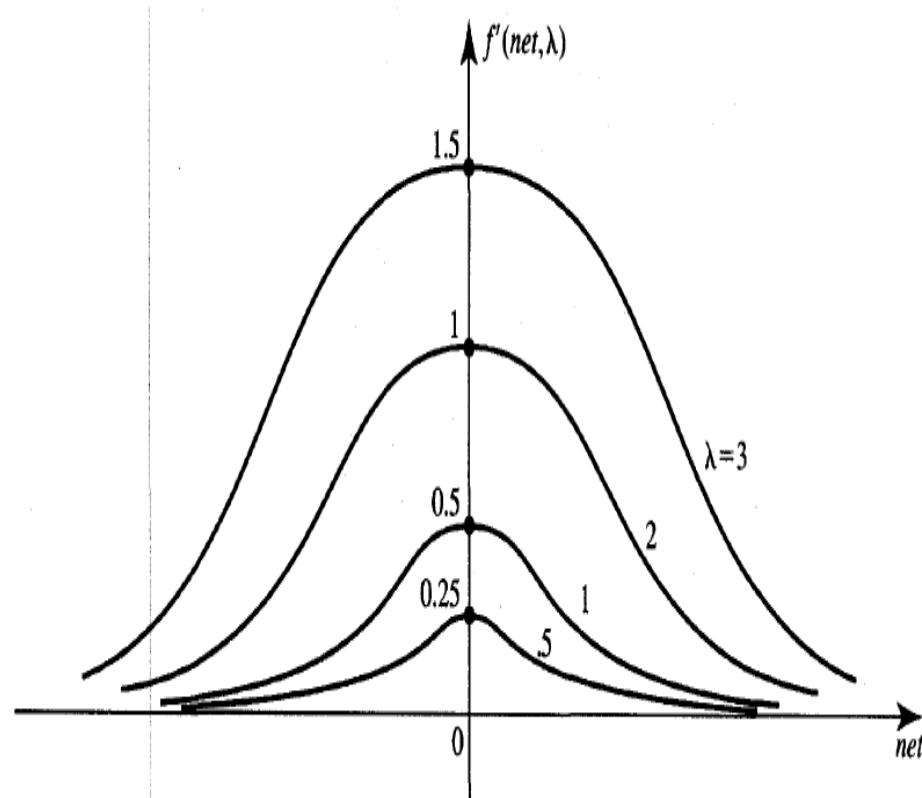
# Steepness of the Activation Function (Steepness factor)

- Determines slope of activation function
- Depends on value of net
- multiplying factor in building components of the error signal vectors  $\delta_o$ , and  $\delta_y$
- both the choice and shape of the activation function strongly affect the speed of network learning
- derivative of the activation function-

$$f'(net) = \frac{2\lambda \exp(-\lambda net)}{[1 + \exp(-\lambda net)]^2}$$

- reaches a maximum value of  $1/2\lambda$  at  $net = 0$

# Steepness of the Activation Function



- For a fixed learning constant all adjustments of weights in proportion to the steepness coefficient  $\lambda$
- Using activation functions with large  $\lambda$  may yield results similar as in the case of large learning constant
- Keep  $\lambda$  at a standard value of 1, and control the learning speed using solely the learning constant  $\eta$ , rather than controlling both  $\lambda$  and  $\eta$

# Learning Constant

- choice of the learning constant  $\rightarrow$  depends strongly on the class of the learning problem and on the network architecture
- values ranging from  $10^{-3}$  to 10 reported as successful
- For large learning constants, the learning speed can be drastically increased

# Momentum Method (Factor)

- To accelerate the convergence of the error back-propagation learning algorithm

$$\Delta w(t) = -\eta \nabla E(t) + \alpha \Delta w(t-1) \quad \leftarrow \text{momentum term}$$

- $\alpha$  is chosen between 0.1 and 0.8