Al-Mustaqbal University

College of Engineering

Department of Biomedical Engineering

Stage: 5th Subject : Neural Eng.

2025-2026

Lecture ( ):

6.2.2 The Backpropagation Algorithm:

Recall from section 2.2.2 the three-layer network in abbreviated notation is shown in Figure 37.

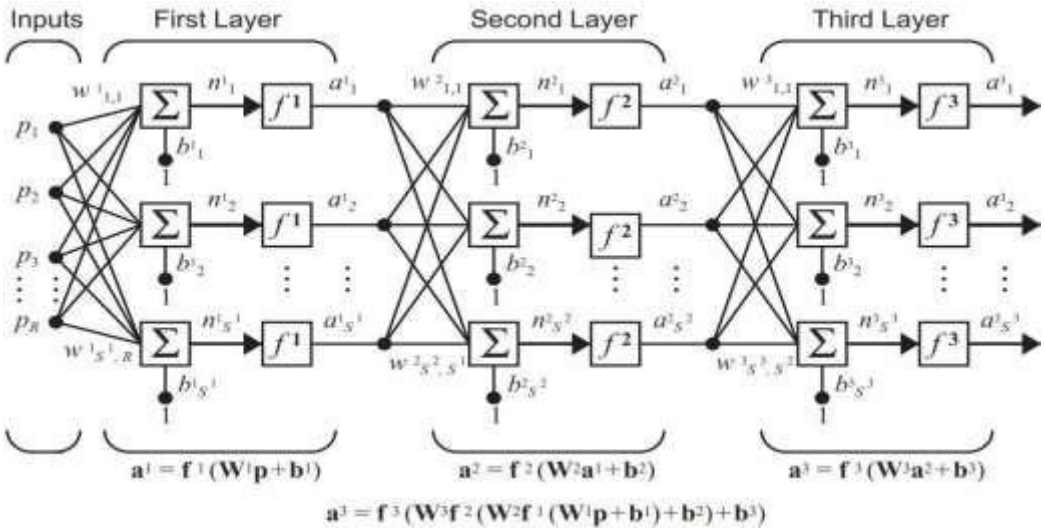


Figure 37: Three-Layer Network

As we discussed earlier, for multilayer networks the output of one layer becomes the input to the following layer. The equations that describe this operation are

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\,\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad for\ m = 0,1,\dots,M-1 \qquad (6.6)$$

where M is the number of layers in the network.

6.2.2.1 The Algorithm:

Step 1: Propagate the input forward through the network:

$$\mathbf{a}^0 = \mathbf{p}$$

Step 2: Compute the output for each layer

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\,\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad for\ m = 0,1,\dots,M-1$$

The outputs of the neurons in the last layer are considered the network outputs:

$$\mathbf{a} = \mathbf{a}^M$$

Step 3: Propagate the sensitivities backward through the network:

$$(\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \cdots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1)$$

$$\mathbf{s}^M = -2\,\mathbf{F}'^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})$$

$$\mathbf{s}^m = \mathbf{F}'^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\,\mathbf{s}^{m+1} \qquad for \quad m = M-1,\ldots,2,1.$$

Where

$$\mathbf{F}'^m(\mathbf{n}^m) = \begin{bmatrix} f'^m(n^m_1) & 0 & \ldots & 0 \\ 0 & f'^m(n^m_2) & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & f'^m(n^m_{s^m}) \end{bmatrix}$$

Step 4: Update the weights and biases are updated using the approximate steepest descent rule:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\,\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\,\mathbf{s}^m$$

6.2.2.2 Example:

Consider the following network shown in Figure 38:

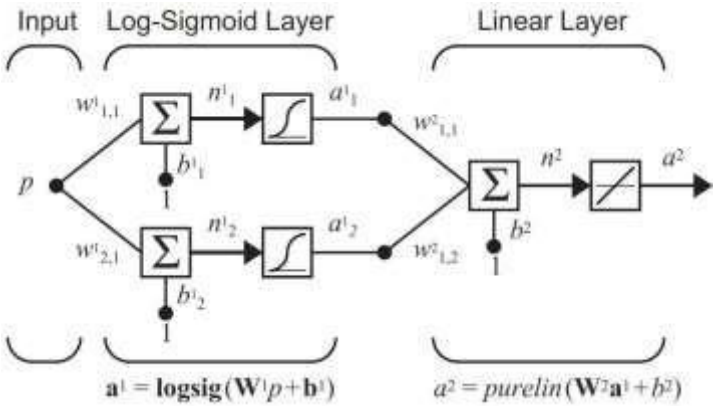


Figure 38: Function Approximation Network

Use the backpropagation algorithm to train the above network, where the initial values for the network weights and biases.

$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}, \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}, \mathbf{W}^2(0) = [0.09 \quad -0.17], \mathbf{b}^2(0) = [0.48]$$

Also, the input is $p = 1$ and the output is $t = 1.707$.

Solution:

Let $a^0 = p = 1$

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1 a^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}[1] + \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix} -0.75 \\ -0.54 \end{bmatrix}\right)$$

$$= \begin{bmatrix} \frac{1}{1+e^{0.75}} \\ \frac{1}{1+e^{0.54}} \end{bmatrix} = \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix}$$

The second layer output is

$$a^2 = f^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) = purelin\left([0.09 \quad -0.17]\begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix} + [0.48]\right) = [0.446]$$

The error would then be

$$e = t - a = t - a^2 = 1.707 - 0.446 = 1.261.$$

The next stage of the algorithm is to backpropagate the sensitivities. Before we begin the backpropagation, recall that we will need the derivatives of the transfer functions, $f'^1(n)$ and $f'^2(n)$. For the first layer

$$f'^1(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1 - a^1)(a^1).$$

For the second layer we have

$$f'^2(n) = \frac{d}{dn}(n) = 1$$

We can now perform the backpropagation. The starting point is found at the second layer, using:

$$s^2 = -2\,\dot{F}^2(n^2)(t-a) = -2[\dot{f}^2(n^2)](1.261) = -2[1](1.261) = -2.522.$$

The first layer sensitivity is then computed by backpropagating the sensitivity from the second layer, using:

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T s^2 = \begin{bmatrix} (1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1) \end{bmatrix} \begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix} [-2.522]$$

$$= \begin{bmatrix} (1-0.321)(0.321) & 0 \\ 0 & (1-0.368)(0.368) \end{bmatrix} \begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix} [-2.522]$$

$$= \begin{bmatrix} 0.218 & 0 \\ 0 & 0.233 \end{bmatrix} \begin{bmatrix} -0.227 \\ 0.429 \end{bmatrix} = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}$$

The final stage of the algorithm is to update the weights. For simplicity, we will use a learning rate $\alpha = 0.1$. We have

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha s^2(\mathbf{a}^1)^T = [0.09 \;\; -0.17] - (0.1)[-2.522][0.321\;\; 0.368]$$

$$= [0.171 \;\; -0.0772]$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha s^2 = [0.48] - (0.1)[-2.522] = [0.732]$$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha \mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} - (0.1) \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} [1] = \begin{bmatrix} -0.265 \\ -0.420 \end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha \mathbf{s}^1 = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} - (0.1) \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} = \begin{bmatrix} -0.475 \\ -0.140 \end{bmatrix}$$

This completes the first iteration of the backpropagation algorithm. We next proceed

to randomly choose another input from the training set and perform another iteration of the algorithm. We continue to iterate until the difference between the network response and the target function reaches some acceptable level.

6.2.2.3 Example:

Use Backpropagation Algorithm to train the following network shown in Figure 39:

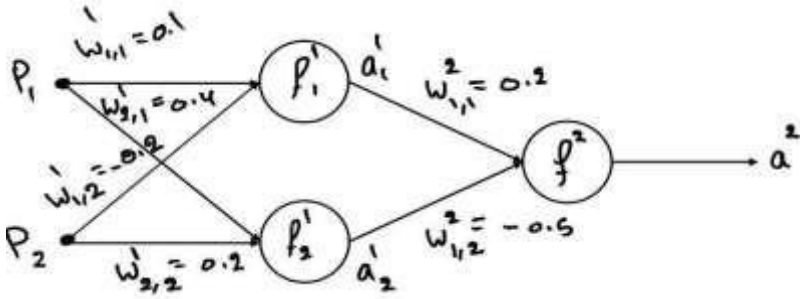


Figure 39: Neural Network

Where $\mathbf{f}^1$ & f^2 are sigmoid functions, with the initial weights

$$\mathbf{W}^1(0) = \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix}, \quad \mathbf{W}^2(0) = [0.2 \quad -0.5]$$

And, the training set $\{\mathbf{p_1}, t_1\}, \{\mathbf{p_2}, t_2\}, \dots, \{\mathbf{p_5}, t_5\}$ are

p_1	p_2	t
0.4	−0.7	0.1
0.3	−0.5	0.05
0.6	0.1	0.3
0.2	0.4	0.25
0.1	−0.2	0.12

Solution:

Let $\mathbf{a}^0 = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}$

The first layer output is

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\boldsymbol{a}^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix}\begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix} 0.18 \\ 0.02 \end{bmatrix}\right)$$

$$= \begin{bmatrix} \frac{1}{1+e^{-0.18}} \\ \frac{1}{1+e^{-0.02}} \end{bmatrix} = \begin{bmatrix} 0.5448 \\ 0.505 \end{bmatrix}$$

The second layer output is

$$a^2 = f^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) = logsig\ ([0.2 \quad -0.5]\begin{bmatrix}0.5448\\0.505\end{bmatrix}) = logsig([-0.14354])$$

$$= [0.4642]$$

The error would then be

$$e = t - a = t - a^2 = 0.1 - 0.4642 = -0.3642.$$

We can now perform the backpropagation. The second layer sensitivity is

$$s^2 = -2\ [f'^2(n^2)](t - a) = -2(1 - a^2)a^2(-0.3642)$$

$$= -2(1 - 0.4642)(0.4642)(-0.3642) = -2(0.09058) = -0.18116.$$

The first layer sensitivity is

$$\mathbf{s}^1 = \mathbf{F}'^1(\mathbf{n}^1)(\mathbf{W}^2)^T\ s^2 = \begin{bmatrix}(1 - a^1_1)(a^1_1) & 0\\ 0 & (1 - a^1_2)(a^1_2)\end{bmatrix}\begin{bmatrix}0.2\\-0.5\end{bmatrix}[-0.18116]$$

$$= \begin{bmatrix}(1 - 0.5448)(0.5448) & 0\\ 0 & (1 - 0.505)(0.505)\end{bmatrix}\begin{bmatrix}0.2\\-0.5\end{bmatrix}[-0.18116]$$

$$= \begin{bmatrix}0.248 & 0\\ 0 & 0.25\end{bmatrix}\begin{bmatrix}0.2\\-0.5\end{bmatrix}[-0.18116] = \begin{bmatrix}0.0496\\-0.125\end{bmatrix}[-0.18116] = \begin{bmatrix}-0.009\\0.023\end{bmatrix}$$

Now, update the weights with learning rate $\alpha = 0.6$.

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha s^2(\mathbf{a}^1)^T = [0.2 \quad -0.5] - (0.6)[-0.18116][0.5448 \quad 0.505]$$

$$= [0.2 \quad -0.5] - 0.109[0.5448 \quad 0.505]$$

$$= [0.2 \quad -0.5] + [0.059 \quad 0.055]$$

$$= [0.259 \quad -0.028]$$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix}0.1 & -0.2\\ 0.4 & 0.2\end{bmatrix} - (0.6)\begin{bmatrix}-0.009\\0.023\end{bmatrix}[0.4 \quad -0.7]$$

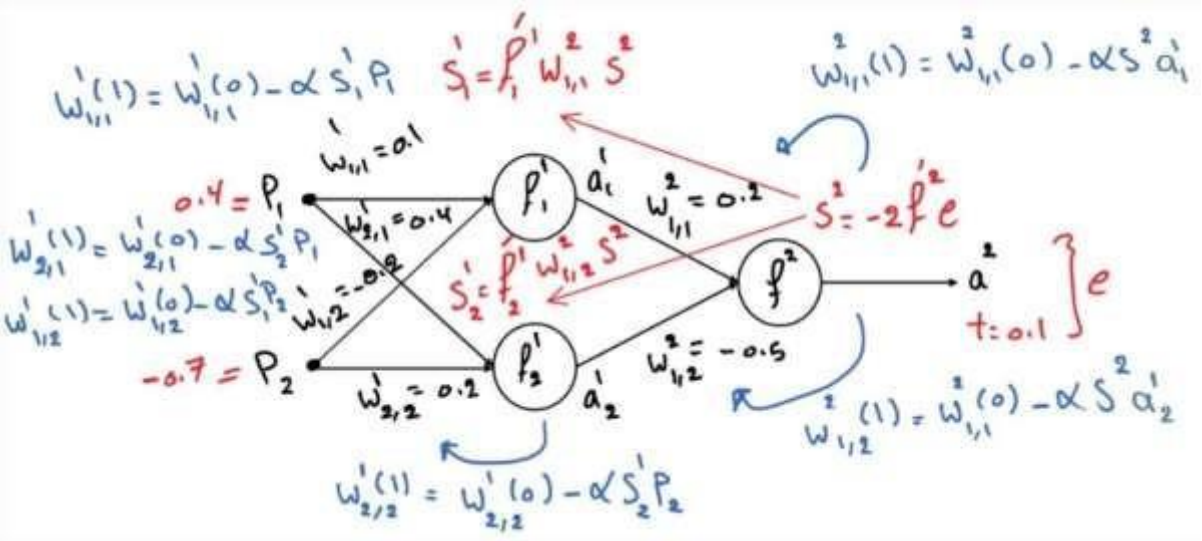
$$= \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix} - (0.6) \begin{bmatrix} -0.004 & 0.006 \\ 0.009 & -0.161 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix} - \begin{bmatrix} -0.0024 & 0.0036 \\ 0.0054 & -0.0966 \end{bmatrix} = \begin{bmatrix} 0.098 & -0.204 \\ 0.395 & 0.497 \end{bmatrix}$$

This completes the first iteration of the backpropagation algorithm. We next proceed to randomly choose another input from the training set and perform another iteration of the algorithm. We continue to iterate until the difference between the network response and the target function reaches some acceptable level.

6.2.2.4 Summary:

The above example 6.2.2.3 can be summarize in following Figure:



7. Hopfield Network:

7.1 Discrete Hopfield Network:

The Hopfield neural network is perhaps the simplest type of neural network. As shown in Figure 40, a Hopfield network consists of a single layer of neurons, 1, 2, . . . , *s*. The network is **fully interconnected**; that is, every neuron in the network is connected to every other neuron. The network is **recurrent**; that is, it has **feedforward/feedbackward** capabilities, which means input to the neurons comes from external input as well as from the neurons themselves internally. Also, the network is autoassociative which means that if the neural network recognizes a pattern, it will return that pattern (i.e. input **p** = target **t**).

The Hopfield network is classified under supervised learning.

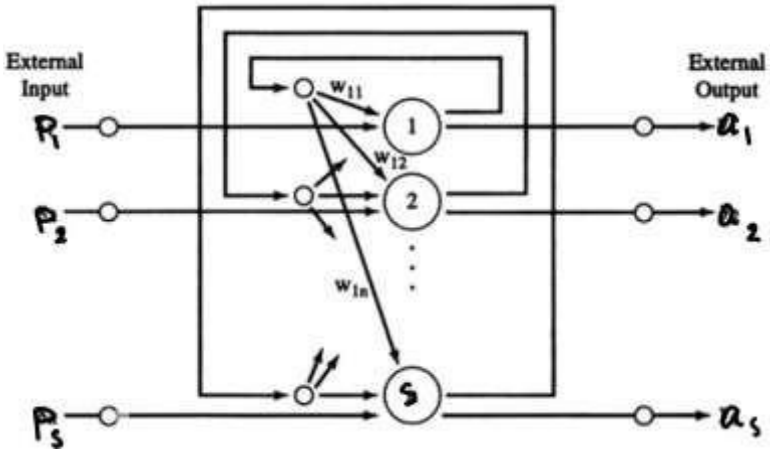


Figure 40: Hopfield Neural Network

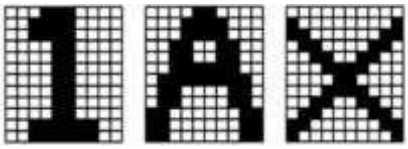


Figure 41: Sample Pattern

Each input/output, p_i *or* a_j, takes a discrete bipolar value of either 1 or −1. The number of neurons, s, is the size required for each pattern in the bipolar representation. For example, suppose that each pattern is a letter represented by an 10 × 12 two-dimensional array, where each array element is either 1 for a black square or −1 for a blank square (for example, Figure 41). Then s will be $10 \times 12 = 120$. Each neuron is associated by weight, w_{ij}, which satisfies the following conditions:

$w_{ij} = w_{ji}$ $\quad for\ all\ \ i, j = 1, 2, \dots, s$ (i.e. **W** is symmetric)

$w_{ii} = 0$ $\quad for\ all\ \ i = 1, 2, \dots, s$ (i.e. There is no self-feedback)

So, the weight matrix is

$$\mathbf{W} = \begin{bmatrix} 0 & w_{1,2} & w_{1,3} & \dots & w_{1,s} \\ w_{2,1} & 0 & w_{2,3} & \cdots & w_{2,s} \\ \vdots & & & \$ & \vdots \\ w_{S,1} & w_{S,2} & w_{S,3} & \cdots & 0 \end{bmatrix}$$