CHAPTER 4
Multilayer Perceptrons

Problem 4.1

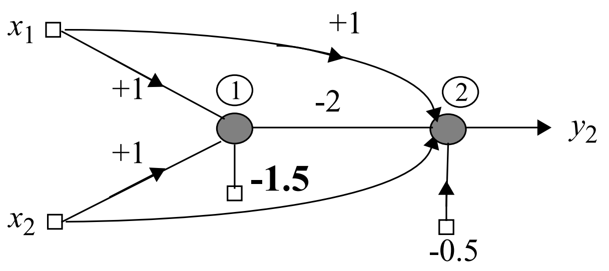

Figure 4: Problem 4.1

Assume that each neuron is represented by a McCulloch-Pitts model. Also assume that

$$x_i = \begin{cases} 1 & \text{if the input bit is 1} \\ 0 & \text{if the input bit is 0} \end{cases}$$

The induced local field of neuron 1 is

$$v_1 = x_1 + x_2 - 1.5$$

We may thus construct the following table:

x_1	0	0	1	1
x_2	0	1	0	1
v_1	-1.5	-0.5	-0.5	0.5
y_2	0	0	0	1

The induced local field of neuron 2 is

$$v_2 = x_1 + x_2 - 2y_1 - 0.5$$

Accordingly, we may construct the following table:

x_1	0	0	1	1
x_2	0	1	0	1
y_1	0	0	0	1
v_2	-0.5	0.5	-0.5	-0.5
y_2	0	1	1	1

From this table we observe that the overall output y_2 is 0 if x_1 and x_2 are both 0 or both 1, and it is 1 if x_1 is 0 and x_2 is 1 or vice versa. In other words, the network of Fig. P4.1 operates as an EXCLUSIVE OR gate.

Problem 4.2

Figure 1 shows the evolutions of the free parameters (synaptic weights and biases) of the neural network as the back-propagation learning process progresses. Each epoch corresponds to 100 iterations. From the figure, we see that the network reaches a steady state after about 25 epochs. Each neuron uses a logistic function for its sigmoid nonlinearity. Also the desired response is defined as

$$d = \begin{cases} 0.9 & \text{for symbol (bit) 1} \\ 0.1 & \text{for symbol (bit) 0} \end{cases}$$

Figure 2 shows the final form of the neural network. Note that we have used biases (the negative of thresholds) for the individual neurons.

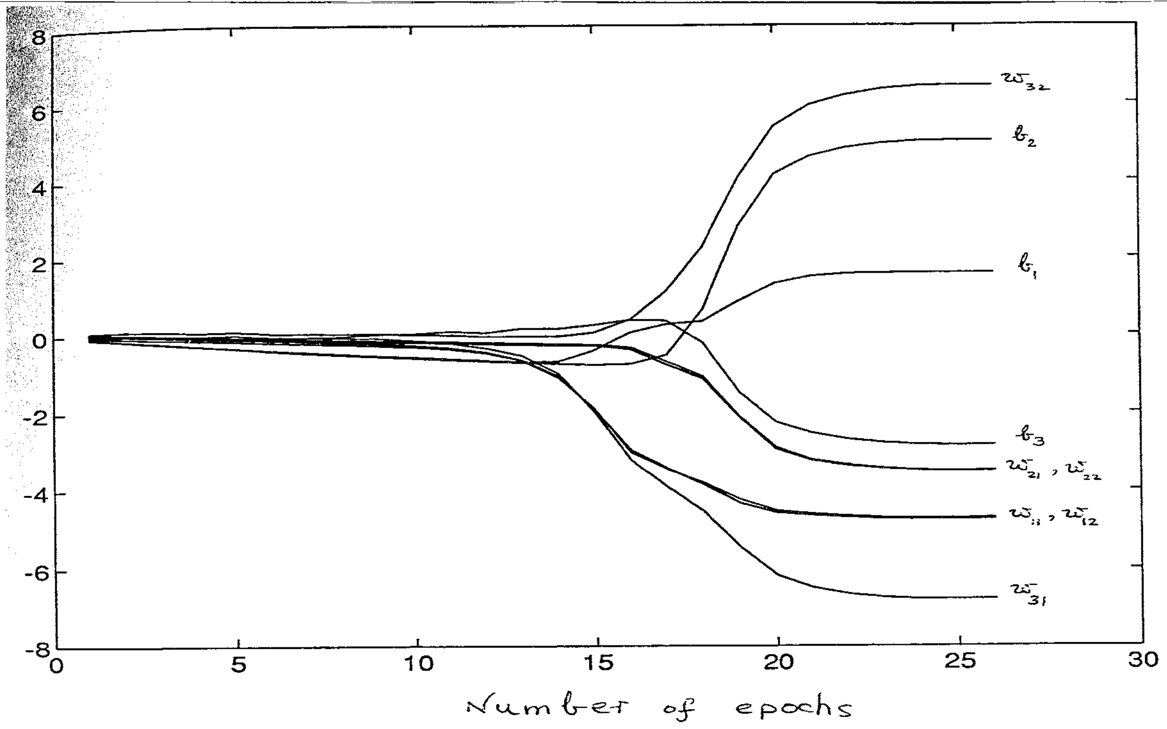

Figure 1: Problem 4.2, where one epoch = 100 iterations

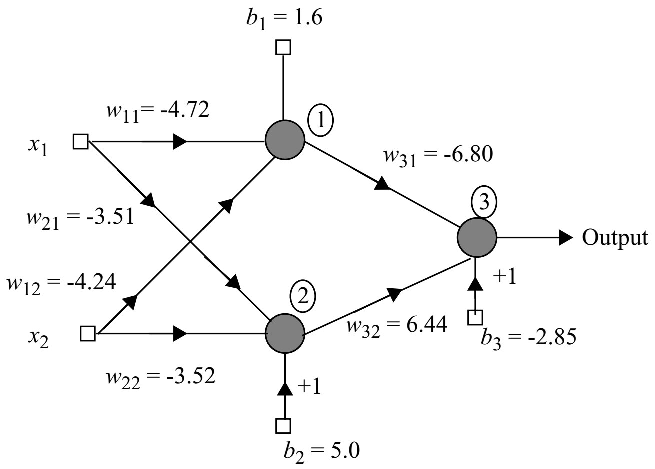

Figure 2: Problem 4.2

Problem 4.3

If the momentum constant α is negative, Equation (4.43) of the text becomes

$$\begin{aligned}\Delta w_{ji}(n) &= -\eta\sum_{t=0}^{n}\alpha^{n-t}\frac{\partial E(t)}{\partial w_{ji}(t)}\\ &= -\eta\sum_{t=0}^{n}(-1)^{n-t}|\alpha|^{n-t}\frac{\partial E(t)}{\partial w_{ji}(t)}\end{aligned}$$

Now we find that if the derivative $\partial E/\partial w_{ji}$ has the same algebraic sign on consecutive iterations of the algorithm, the magnitude of the exponentially weighted sum is reduced. The opposite is true when $\partial E/\partial w_{ji}$ alternates its algebraic sign on consecutive iterations. Thus, the effect of the momentum constant α is the same as before, except that the effects are reversed, compared to the case when α is positive.

Problem 4.4

From Eq. (4.43) of the text we have

$$\Delta w_{ji}(n) = -\eta\sum_{t=1}^{n}\alpha^{n-t}\frac{\partial E(t)}{\partial w_{ji}(t)} \tag{1}$$

For the case of a single weight, the cost function is defined by

$$E = k_1(w-w_0)^2 + k_2$$

Hence, the application of (1) to this case yields

$$\Delta w(n) = -2k_1\eta\sum_{t=1}^{n}\alpha^{n-t}(w(t)-w_0)$$

In this case, the partial derivative $\partial E(t)/\partial w(t)$ has the same algebraic sign on consecutive iterations. Hence, with $0 \le \alpha < 1$ the exponentially weighted adjustment $\Delta w(n)$ to the weight w at time n grows in magnitude. That is, the weight w is adjusted by a large amount. The inclusion of the momentum constant α in the algorithm for computing the optimum weight $w^* = w_0$ tends to *accelerate* the downhill descent toward this optimum point.

Problem 4.5

Consider Fig. 4.14 of the text, which has an input layer, two hidden layers, and a single output neuron. We note the following:

$$y_1^{(3)} = F(A_1^{(3)}) = F(\mathbf{w}, \mathbf{x})$$

Hence, the derivative of $F(A_1^{(3)})$ with respect to the synaptic weight $w_{1k}^{(3)}$ connecting neuron k in the second hidden layer to the single output neuron is

$$\frac{\partial F(A_1^{(3)})}{\partial w_{1k}^{(3)}} = \frac{\partial F(A_1^{(3)})}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial v_1^{(3)}}\frac{\partial v_1^{(3)}}{\partial w_{1k}^{(3)}} \tag{1}$$

where $v_1^{(3)}$ is the activation potential of the output neuron. Next, we note that

$$\frac{\partial F(A_1^{(3)})}{\partial y_1^{(3)}} = 1$$

$$y_1^{(3)} = \varphi(v_1^{(3)})$$

$$v_1^{(3)} = \sum_k w_{1k}^{(3)} y_k^{(2)} \tag{2}$$

where $y_k^{(2)}$ is the output of neuron k in layer 2. We may thus proceed further and write

$$\frac{\partial y_1^{(3)}}{\partial v_1^{(3)}} = \varphi'(v_1^{(3)}) = \varphi' A_1^{(3)} \tag{3}$$

$$\frac{\partial v_1^{(3)}}{\partial w_{1k}^{(3)}} = y_k^{(2)}$$

$$= \varphi(A_k^{(2)}) \qquad (4)$$

Thus, combining (1) to (4):

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{1k}^{(3)}} = \frac{\partial F(A_1^{(3)})}{\partial w_{1k}^{(3)}}$$

$$= \varphi'(A_1^{(3)})\varphi(A_k^{(3)})$$

Consider next the derivative of $F(\mathbf{w,x})$ with respect to $w_{kj}^{(2)}$, the synaptic weight connecting neuron *j* in layer 1 (i.e., first hidden layer) to neuron k in layer 2 (i.e., second hidden layer):

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{kj}^{(2)}} = \frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial y_1^{(3)}} \frac{\partial y_1^{(3)}}{\partial v_1^{(3)}} \frac{\partial v_1^{(3)}}{\partial y_k^{(2)}} \frac{\partial y_k^{(2)}}{\partial v_k^{(2)}} \frac{\partial v_k^{(2)}}{\partial w_{kj}^{(2)}} \qquad (5)$$

where $y_k^{(2)}$ is the output of neuron in layer 2, and $v_k^{(1)}$ is the activation potential of that neuron. Next we note that

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial y_1^{(3)}} = 1 \qquad (6)$$

$$\frac{\partial y_1^{(3)}}{\partial v_1^{(3)}} = \varphi'(A_1^{(3)}) \qquad (7)$$

$$v_1^{(3)} = \sum_k w_{1k}^{(3)} y_k^{(2)}$$

$$\frac{\partial v_1^{(3)}}{\partial y_k^{(2)}} = w_{1k}^{(3)} \qquad (8)$$

$$y_k^{(2)} = \varphi(v_k^{(2)})$$

$$\frac{\partial y_k^{(2)}}{\partial v_k^{(2)}} = \varphi'(v_k^{(2)}) = \varphi'(A_k^{(2)}) \qquad (9)$$

$$v_k^{(2)} = \sum_j w_{kj}^{(1)} y_j^{(1)}$$

$$\frac{\partial v_k^{(2)}}{\partial w_{kj}^{(1)}} = y_j^{(1)} = \varphi(v_j^{(1)}) = \varphi(A_j^{(1)}) \tag{10}$$

Substituting (6) and (10) into (5), we get

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{kj}^{(2)}} = \varphi'(A_1^{(3)}) w_{1k}^{(3)} \varphi'(A_k^{(2)}) \varphi(A_j^{(1)})$$

Finally, we consider the derivative of $F(\mathbf{w},\mathbf{x})$ with respect to $w_{ji}^{(1)}$, the synaptic weight connecting source node *i* in the input layer to neuron *j* in layer 1. We may thus write

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{ji}^{(1)}} = \frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial y_1^{(3)}} \frac{\partial y_1^{(3)}}{\partial v_1^{(3)}} \frac{\partial v_1^{(3)}}{\partial y_j^{(1)}} \frac{\partial y_j^{(1)}}{\partial v_j^{(1)}} \frac{\partial v_j^{(1)}}{\partial w_{ji}^{(1)}} \tag{11}$$

where $y_j^{(1)}$ is the output of neuron *j* in layer 1, and $v_i^{(1)}$ is the activation potential of that neuron. Next we note that

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial y_1^{(3)}} = 1 \tag{12}$$

$$\frac{\partial y_1^{(3)}}{\partial v_1^{(3)}} = \varphi'(A^{(3)}) \tag{13}$$

$$v_1^{(3)} = \sum_k w_{1k}^{(3)} y_k^{(2)}$$

$$\begin{aligned} \frac{\partial v_1^{(3)}}{\partial y_j^{(1)}} &= \sum_k w_{1k}^{(3)} \frac{\partial y_k^{(2)}}{\partial y_j^{(1)}} \\ &= \sum_k w_{1k}^{(3)} \frac{\partial y_k^{(2)}}{\partial v_k^{(2)}} \frac{\partial v_k^{(2)}}{\partial y_j^{(1)}} \\ &= \sum_k w_{1k}^{(3)} \varphi'(A_k^{(2)}) \frac{\partial v_k^{(2)}}{\partial y_j^{(1)}} \end{aligned} \tag{14}$$

$$\frac{\partial v_k^{(2)}}{\partial y_j^{(1)}} = w_{kj}^{(2)} \tag{15}$$

$$y_j^{(1)} = \varphi(v_j^{(1)})$$

$$\frac{\partial y_j^{(1)}}{\partial v_j^{(1)}} = \varphi'(v_j^{(1)}) = \varphi'(A_j^{(1)}) \tag{16}$$

$$v_j^{(1)} = \sum_i w_{ji}^{(1)} x_i$$

$$\frac{\partial v_j^{(1)}}{\partial w_{ji}^{(1)}} = x_i \tag{17}$$

Substituting (12) to (17) into (11) yields

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{ji}^{(1)}} = \varphi'(A_1^{(3)})\left(\sum_k w_{1k}^{(3)} \varphi'(A_k^{(2)}) w_{kj}^{(2)}\right)\varphi'(A_j^{(1)}) x_i$$

Problem 4.12

According to the conjugate-gradient method, we have

$$\begin{aligned}\Delta\mathbf{w}(n) &= \eta(n)\mathbf{p}(n)\\ &= \eta(n)[-\mathbf{g}(n) + \beta(n-1)\mathbf{p}(n-1)]\\ &\approx -\eta(n)\mathbf{g}(n) + \beta(n-1)\eta(n-1)\mathbf{p}(n-1)\end{aligned} \tag{1}$$

where, in the second term of the last line in (1), we have used $\eta(n - 1)$ in place of $\eta(\eta)$. Define

$$\Delta\mathbf{w}(n-1) = \eta(n-1)\mathbf{p}(n-1)$$

We may then rewrite (1) as

$$\Delta\mathbf{w}(n) \approx \quad -\eta(n)\mathbf{g}(n) + \beta(n-1)\Delta\mathbf{w}(n-1) \tag{2}$$

On the other hand, according to the generalized delta rule, we have for neuron *j*:

$$\Delta\mathbf{w}_j(n) = \alpha\Delta\mathbf{w}_j(n-1) + \eta\delta_j(n)\mathbf{y}(n) \tag{3}$$

Comparing (2) and (3), we observe that they have a similar mathematical form:

- The vector -$\mathbf{g}(n)$ in the conjugate gradient method plays the role of $\delta_j(n)\mathbf{y}(n)$, where $\delta_j(n)$ is the local gradient of neuron j and $\mathbf{y}(n)$ is the vector of inputs for neuron j.
- The time-varying parameter $\beta(n - 1)$ in the conjugate-gradient method plays the role of momentum α in the generalized delta rule.

Problem 4.13

We start with (4.127) in the text:

$$\beta(n) = -\frac{\mathbf{s}^T(n-1)\mathbf{A}\mathbf{r}(n)}{\mathbf{s}^T(n-1)\mathbf{A}\mathbf{s}(n-1)} \tag{1}$$

The residual $\mathbf{r}(n)$ is governed by the recursion:

$$\mathbf{r}(n) = \mathbf{r}(n-1) - \eta(n-1)\mathbf{A}\mathbf{s}(n-1)$$

Equivalently, we may write

$$-\eta(n-1)\mathbf{A}\mathbf{s}(n-1) = \mathbf{r}(n) - \mathbf{r}(n-1) \tag{2}$$

Hence multiplying both sides of (2) by $\mathbf{s}^T(n - 1)$, we obtain

$$\begin{aligned}\eta(n-1)\mathbf{s}^T(n-1)\mathbf{A}\mathbf{s}(n-1) &= -\mathbf{s}^T(n-1)(\mathbf{r}(n) - \mathbf{r}(n-1)) \\ &= \mathbf{s}^T(n-1)\mathbf{r}(n-1) \end{aligned} \tag{3}$$

where it is noted that (by definition)

$$\mathbf{s}^T(n-1)\mathbf{r}(n) = 0$$

Moreover, multiplying both sides of (2) by $\mathbf{r}^T(n)$, we obtain

$$\begin{aligned}-\eta(n-1)\mathbf{r}^T(n)\mathbf{A}\mathbf{s}(n-1) &= -\eta(n-1)\mathbf{s}^T(n-1)\mathbf{A}\mathbf{r}(n-1) \\ &= \mathbf{r}^T(n)(\mathbf{r}(n) - \mathbf{r}(n-1)) \end{aligned} \tag{4}$$

where it is noted that $\mathbf{A}^T = \mathbf{A}$. Dividing (4) by (3) and invoking the use of (1):

$$\beta(n) = \frac{\mathbf{r}^T(n)(\mathbf{r}(n) - \mathbf{r}(n-1))}{\mathbf{s}^T(n-1)\mathbf{r}(n-1)} \tag{5}$$

which is the Hesteness-Stiefel formula.

In the linear form of conjugate gradient method, we have

$$\mathbf{s}^T(n-1)\mathbf{r}(n-1) = \mathbf{r}^T(n-1)\mathbf{r}(n-1)$$

in which case (5) is modified to

$$\beta(n) = \frac{\mathbf{r}^T(n)(\mathbf{r}(n)-\mathbf{r}(n-1))}{\mathbf{r}^T(n-1)\mathbf{r}(n-1)} \tag{6}$$

which is the Polak-Ribiére formula. Moreover, in the linear case we have

$$\mathbf{r}^T(n)\mathbf{r}(n-1) = 0$$

in which case (6) reduces to the Fletcher-Reeves formula:

$$\beta(n) = \frac{\mathbf{r}^T(n)\mathbf{r}(n)}{\mathbf{r}^T(n-1)\mathbf{r}(n-1)}$$

Problem 4.15

In this problem, we explore the operation of a fully connected multilayer perceptron trained with the back-propagation algorithm. The network has a single hidden layer. It is trained to realize the following one-to-one mappings:

(a) <u>Inversion</u>:

$$f(x) = \frac{1}{x}, \qquad 1 \le x \le 100$$

(b) <u>Logarithmic computation</u>

$$f(x) = \log_{10}x, \qquad 1 \le x \le 10$$

(c) <u>Exponentiation</u>

$$f(x) = e^{-x}, \qquad 1 \le x \le 10$$

(d) Sinusoidal computation

$$f(x) = \sin x, \qquad 0 \le x \le \frac{\pi}{2}$$

(a) $f(x) = 1/x$ for $1 \le x \le 100$

The network is trained with:

learning-rate parameter $\eta = 0.3$, and
momentum constant $\alpha = 0.7$.

Ten different network configurations were trained to learn this mapping. Each network was trained identically, that is, with the same η and α, with bias terms, and with 10,000 passes of the training vectors (with one exception noted below). Once each network was trained, the test dataset was applied to compare the performance and accuracy of each configuration. Table 1 summarizes the results obtained:

Table 1

Number of hidden neurons	Average percentage error at the network output
3	4.73%
4	4.43
5	3.59
7	1.49
10	1.12
15	0.93
20	0.85
30	0.94
100	0.9
30 (trained with 100,000 passes)	0.19

The results of Table 1 indicate that even with a small number of hidden neurons, and with a relatively small number of training passes, the network is able to learn the mapping described in (a) quite well.

(b) $f(x) = \log_{10} x$ for $1 \leq x \leq 10$

The results of this second experiment are presented in Table 2:

Table 2

Number of hidden neurons	Average percentage error at the network output
2	2.55%
3	2.09
4	0.46
5	0.48
7	0.85
10	0.42
15	0.85
20	0.96
30	1.26
100	1.18
30 (trained with 100,000 passes)	0.41

Here again, we see that the network performs well even with a small number of hidden neurons. Interestingly, in this second experiment the network peaked in accuracy with 10 hidden neurons, after which the accuracy of the network to produce the correct output started to decrease.

(c) $f(x) = e^{-x}$ for $1 \leq x \leq 10$

The results of this third experiment (using the logistic function as with experiments (a) and (b)), are summarized in Table 3:

Table 3

Number of hidden neurons	Average percentage error at the network output
2	244.0‘%
3	185.17
4	134.85
5	133.67
7	141.65
10	158.77
15	151.91
20	144.79
30	137.35
100	98.09
30 (trained with 100,000 passes)	103.99

These results are unacceptable since the network is unable to generalize when each neuron is driven to its limits.

The experiment with 30 hidden neurons and 100,000 training passes was repeated, but this time the hyperbolic tangent function was used as the nonlinearity. The result obtained this time was an average percentage error of 3.87% at the network output. This last result shows that the hyperbolic tangent function is a better choice than the logistic function as the sigmoid function for realizing the mapping $f(x) = e^{-x}$.

(d) $f(x) = \sin x$ for $0 \leq x \leq \pi/2$

Finally, the following results were obtained using the logistic function with 10,000 training passes, except for the last configuration:

Table 4

Number of hidden neurons	Average percentage error at the network output
2	1.63‘%
3	1.25
4	1.18
5	1.11
7	1.07
10	1.01
15	1.01
20	0.72
30	1.21
100	3.19
30 (trained with 100,000 passes)	0.4

The results of Table 4 show that the accuracy of the network peaks around 20 neurons, where after the accuracy decreases.