Chapter 6

Multilayer neural networks

Problem Solutions

Section 6.2

1. Consider a three-layer network with linear units throughout, having input vector $\mathbf{x}$, vector at the hidden units $\mathbf{y}$, and output vector $\mathbf{z}$. For such a linear system we have $\mathbf{y} = \mathbf{W}_1\mathbf{x}$ and $\mathbf{z} = \mathbf{W}_2\mathbf{y}$ for two matrices $\mathbf{W}_1$ and $\mathbf{W}_2$. Thus we can write the output as

$$\begin{aligned} \mathbf{z} &= \mathbf{W}_2\mathbf{y} = \mathbf{W}_2\mathbf{W}_1\mathbf{x} \\ &= \mathbf{W}_3\mathbf{x} \end{aligned}$$

for some matrix $\mathbf{W}_3 = \mathbf{W}_2\mathbf{W}_1$. But this equation is the same as that of a two-layer network having connection matrix $\mathbf{W}_3$. Thus a three-layer network with linear units throughout can be implemented by a two-layer network with appropriately chosen connections.

Clearly, a non-linearly separable problem cannot be solved by a three-layer neural network with linear hidden units. To see this, suppose a non-linearly separable problem can be solved by a three-layer neural network with hidden units. Then, equivalently, it can be solved by a two-layer neural network. Then clearly the problem is linearly separable. But, by assumption the problem is only non-linearly separable. Hence there is a contradiction and the above conclusion holds true.

2. Fourier's theorem shows that a three-layer neural network with sigmoidal hidden units can act as a universal approximator. Consider a two-dimensional input and a single output $z(x_1, x_2) = z(\mathbf{x})$. Fourier's theorem states

$$z(\mathbf{x}) \simeq \sum_{f_1} \sum_{f_2} A_{f_1 f_2} \cos(f_1 x_1) \cos(f_2 x_2).$$

(a) Fourier's theorem, as stated above, can be rewritten with the trigonometric identity:

$$\cos(\alpha)\cos(\beta) = \frac{1}{2}\cos(\alpha + \beta) + \frac{1}{2}\cos(\alpha - \beta),$$

to give

$$z(x_1, x_2) \simeq \sum_{f_1} \sum_{f_2} \frac{A_{f_1 f_2}}{z2} \left[\cos(f_1 x_1 + f_2 x_2) + \cos(f_1 x_1 - f_2 x_2)\right].$$

(b) We want to show that $\cos(x)$, or indeed any continuous function, can be approximated by the following linear combination

$$f(x) \simeq f(x_0) + \sum_{i=0}^{n} [f(x_{i+1} - f(x_i)] \left[\frac{\text{Sgn}[x - x_i]}{2}\right].$$

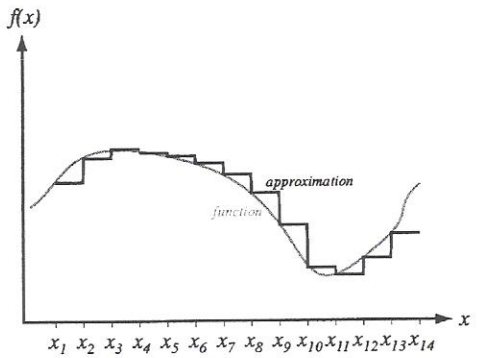
The Fourier series of a function $f(x)$ at a point x_0 converges to $f(x_0)$ if $f(x)$ is of bounded variation in some interval $(x_0 - h, x_0 + h)$ centered on x_0. A function of bounded variation is a follows, given a partition on the interval $a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$ form the sum

$$\sum_{k=1}^{n} |f(x_k) - f(x_{k-1})|.$$

The least upper bound of these sums is called the total variation. For a point $f(x)$ in the neighborhood of $f(x_0)$, we can rewrite the variation as

$$\sum_{i=1}^{n} [f(x_{i+1} - f(x_i)] \left[\frac{\text{Sgn}[x - x_i]}{2}\right],$$

which sets the interval to be $(x, x + 2h)$. Note the function has to be either continuous at $f(x_0)$ or have a discontinuity of the first kind.



(c) As the effective width of the sigmoid vanishes, i.e., as $\sigma \to 0$, the sigmoids become step functions. Then the functions $\cos(f_1 x_1 + f_2 x_2)$ and $\cos(f_1 x_1 - f_2 x_2)$ can be approximated as

$$\begin{aligned}\cos(f_1 x_1 + f_2 x_2) \simeq\ & \cos(f_1 x_{1_0} + f_2 x_{2_0}) \\ & + \left(\sum_{i=0}^{n} [\cos(x_{1_{i+1}} f_1 + x_{2_{i+1}} f_2) - \cos(x_{1_i} f_1 + x_{2_i} f_2)] \right. \\ & \left. \times \left[\frac{\text{Sgn}[x_1 - x_{1i}]\text{Sgn}[x_2 - x_{2i}]}{2}\right] \right),\end{aligned}$$

and similarly for $\cos[f_1 x_1 - f_2 x_2]$.

(d) The construction does not necessarily guarantee that the derivative is approximated, since there might be discontinuities of the first order, that is, non-continuous first derivative. Nevertheless, the one-sided limits of $f(x_0 + 0)$ and $f(x_0 - 0)$ will exist.

Section 6.3

3. Consider a $d - n_H - c$ network trained with n patterns for m_e epochs.

(a) Consider the space complexity of this problem. The total number of adjustable weights is $dn_H + n_Hc$. The amount of storage for the n patterns is nd.

(b) In stochastic mode we choose pattern randomly and compute

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t).$$

until stopping criterion is met. Each iteration involves computing $\Delta\mathbf{w}(t)$ and then adding it to $\mathbf{w}(t)$. From Eq. 17 in the text, for hidden-to-output unit weights we have

$$\Delta w_{jk} = \eta(t_k - z_k)f'(net_k)y_j,$$

where $net_k = \sum_{j=1}^{n_H} w_{jk}y_j$ is computed by n_H multiplications and n_H additions. So, Δw_{jk} is computed with $c(2n_H + 1)$ operations.

From Eq. 17 in the text, for input-to-hidden unit weights we have

$$\Delta w_{ji} = \eta x_i f'(net_k)\sum_{k=1}^{c} w_{kj}\delta_k$$

where, net_j is computed in $2d$ operations. Moreover, $\sum_k w_{jk}\delta_k$ is computed in $2c$ operations. Thus we have w_{ij}'s are computed in $[2d + 2c]n_H$ time.

Thus, the time for one iteration is the time to compute $\Delta\mathbf{w}$ plus the time to add $\mathbf{w}$ to $\Delta\mathbf{w}$, that is,

$$\begin{aligned} T &= c(2n_H + 10 + 2(d + c + 1)n_H + (dn_H + n_Hc) \\ &= 3dn_H + 5n_Hc + c + 2n_H. \end{aligned}$$

In summary, the time complexity is $(3dn_H + 5n_Hc + c + 2n_H)m_e$.

(c) Here, the number of iterations $= nm_e$ and thus the time complexity is $(3dn_H + 5n_Hc + c + 2n_H)n\ m_e$.

4. Equation 20 in the text gives the sensitivity at a hidden unit

$$\delta_j = f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k.$$

For a four-layer or higher-layer neural network, the sensitivity of a hidden unit is likewise given by

$$\delta_j \equiv -\frac{\partial E}{\partial net_j} = -\left[\frac{\partial E}{\partial o_j}\frac{\partial O_j}{\partial net_j}\right].$$

10. We express the derivative of a sigmoid in terms of the sigmoid itself for positive constants a and b for the following cases.

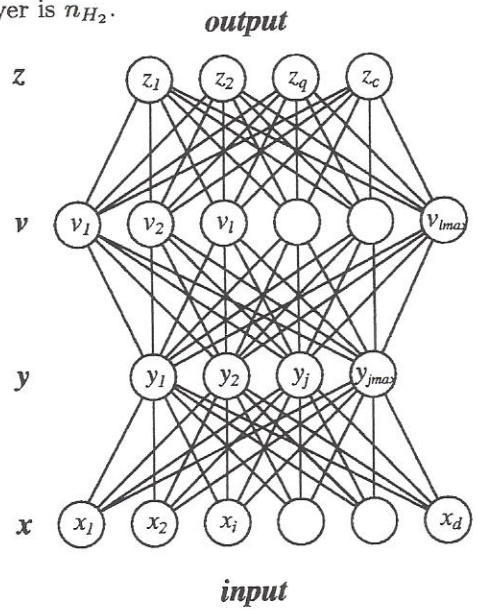
(a) For $f(net) = 1/(1 + e^{a\ net})$, the derivative is

$$\begin{aligned}\frac{df(net)}{d\ net} &= -a\left(\frac{1}{1+e^{a\ net}}\right)^2 e^{a\ net} \\ &= -af(net)(1 - f(net)).\end{aligned}$$

(b) For $f(net) = a\tanh(b\ net)$, the derivative is

$$\frac{df(net)}{d\ net} = -2b^2 a\tanh(b\ net)(1 - \tanh^2(b\ net)).$$

11. We use the following notation: The activations at the first (input), second, third, and fourth (output) layers are x_i, y_j, v_l, and z_k, respectively, and the indexes are clear from usage. The number of units in the first hidden layer is n_{H_1} and the number in the second hidden layer is n_{H_2}.



Algorithm 0 (Four-layer backpropagation)

```
1 begin initialize xxx
2       xxx
3       xxx
4       return xxx
5 end
```

Section 6.4

12. Suppose the input to hidden weights are set equal to the same value, say w_o, then $w_{ij} = w_o$. Then we have

$$net_j = f(net_j) = \sum_{i=1}^{d} w_{ji}x_i = w_o \sum_i x_i = w_o \mathbf{x}.$$

(b) Already shown above.

19. The assumption that the network can represent the underlying true distribution is not used before Eq. 28 in the text. For Eq. 29, however, we invoke $g_k(\mathbf{x};\ \mathbf{w}) \simeq p(\omega_k|\mathbf{x})$, which is used for

$$\sum_{k=1}^{c} \int [g_k(\mathbf{x};\ \mathbf{w}) - P(\omega_k|\mathbf{x})]\, p(\mathbf{x}) d\mathbf{x} = 0.$$

This is true only when the above assumption is met. If the assumption is not met, the gradient descent procedure yields the closest projection to the posterior probability in the class spanned by the network.

20. Recall the equation

$$p(\mathbf{y}|\omega_k) = e^{A(\tilde{\mathbf{w}}_k) + B(\mathbf{y},\phi) + \tilde{\mathbf{w}}^t \mathbf{y}}.$$

(a) Given $p(\mathbf{y}|\omega_k)$, we use Bayes' Theorem to write the posterior as

$$p(\omega_k|\mathbf{y}) = \frac{p(\mathbf{y}|\omega_k) P(\omega_k)}{p(\mathbf{y})}.$$

(b) We interpret $A(\cdot)$, $\tilde{\mathbf{w}}_k$ and ϕ as follows:

$$\begin{aligned} P(\omega_k) &= e^{-A(\tilde{\mathbf{w}}_k)} \\ p(\omega_k|\mathbf{y}) &= \frac{\exp\left[A(\tilde{\mathbf{w}}_k) + B(\mathbf{y},\phi) + \tilde{\mathbf{w}}_k^t \mathbf{y}\right] P(\omega_k)}{\sum_{m=1}^{c} \exp\left[A(\tilde{\mathbf{w}}_m) + B(\mathbf{y},\phi) + \tilde{\mathbf{w}}_m^t \mathbf{y}\right] P(\omega_m)} \\ &= \frac{net_k}{\sum_{m=1}^{c} e^{net_m}}, \end{aligned}$$

where $net_k = b(\mathbf{y},\phi) + \tilde{\mathbf{w}}^t \mathbf{y}$. Thus, $B(\mathbf{y},\phi)$ is the bias, $\tilde{\mathbf{w}}$ is the weight vector describing the separating plane, and $e^{-A(\tilde{\mathbf{w}}_k)}$ is $P(\omega_k)$.

21. Backpropagation with softmax is done in the usual manner; all that must be evaluated differently are the sensitivities at the output and hidden layer units, that is,

$$z_h = \frac{e^{net_h}}{\sum_h e^{net_h}} \qquad \text{and} \qquad \frac{\partial z_h}{\partial net_h} = z_h(1 - z_h).$$

(a) We are given the following terms:

$$\begin{aligned} net_j &= \sum_{i=1}^{d} w_{ji} x_i \\ net_k &= \sum_{j=1}^{n_H} w_{kj} y_j \\ y_j &= f(net_j) \\ z_k &= \frac{e^{net_k}}{\sum_{m=1}^{c} e^{net_m}}, \end{aligned}$$

and the error function

$$J = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2.$$

To derive the learning rule we have to compute $\partial J/\partial w_{kj}$ and $\partial J/\partial w_{ji}$. We start with the former:

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{net_k}\frac{\partial net_k}{\partial w_{kj}}.$$

We next compute

$$\frac{\partial J}{\partial net_k} = \sum_{s=1}^{c}\frac{\partial J}{\partial z_s}\frac{\partial z_s}{\partial net_k}.$$

We also have

$$\frac{\partial z_s}{\partial net_k} = \begin{cases} \frac{e^{net_s}(-1)e^{net_k}}{\left(\sum_{m=1}^{c} e^{net_m}\right)^2} = -z_s z_k & \text{if } s \neq k \\ \frac{e^{net_s}}{\left(\sum_{m=1}^{c} e^{net_m}\right)} + (-1)\frac{e^{net_s}e^{net_s}}{\left(\sum_{m=1}^{c} e^{net_m}\right)^2} = z_k - z_k^2 & \text{if } s = k \end{cases}$$

and finally

$$\frac{\partial J}{\partial z_s} = (-1)(t_s - z_s).$$

Putting these together we get

$$\frac{\partial J}{\partial net_k} = \sum_{s\neq k}^{c}(-1)(t_s - z_s)(-z_s z_k) + (-1)(t_k - z_k)(z_k - z_k^2).$$

We use $\partial net_k/\partial w_{kj} = y_j$ and obtain

$$\frac{\partial J}{\partial w_{kj}} = y_j\sum_{s\neq k}^{c}(t_s - z_s)(z_s z_k) - y_j(t_k - z_k)(z_k - z_k^2).$$

Now we have to find input-to-hidden weight contribution to J. By the chain rule we have

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}.$$

We can at once find out the last two partial derivatives.

$$\frac{\partial y_j}{\partial net_j} = f'(net_j) \text{ and } \frac{\partial net_j}{\partial w_{ji}} = x_i.$$

Now we also have

$$\frac{\partial J}{\partial y_j} = \sum_{s=1}^{c}\frac{\partial J}{\partial z_s}\frac{\partial z_s}{\partial y_j}$$

$$
\begin{aligned}
&= -\sum_{s=1}^{c}(t_s - z_s)\frac{\partial z_s}{\partial y_j} \\
&= -\sum_{s=1}^{c}(t_s - z_s)\left[\sum_{r=1}^{c}\frac{\partial z_s}{\partial net_r}\frac{\partial net_r}{\partial y_j}\right] \\
&= -\sum_{s=1}^{c}(t_s - z_s)\left[\underbrace{\frac{\partial z_s}{\partial net_s}}_{z_s - z_s^2}\underbrace{\frac{\partial net_s}{\partial y_j}}_{w_{sj}} + \sum_{r\neq s}^{c}\underbrace{\frac{\partial z_s}{\partial net_r}}_{-z_s z_r}\underbrace{\frac{\partial net_r}{\partial y_j}}_{w_{rj}}\right] \\
&= -\sum_{s=1}^{c}(t_s - z_s)(z_s - z_s^2)w_{sj} + \sum_{s=1}^{c}\sum_{r\neq s}^{c} z_s z_r w_{rj}(t_s - z_s).
\end{aligned}
$$

We put all this together and find

$$
\begin{aligned}
\frac{\partial J}{\partial w_{ji}} &= x_i f'(net_j)\sum_{s=1}^{c}(t_s - z_s)\sum_{r\neq s}^{c} w_{rj} z_s z_r \\
&\quad - x_i f'(net_j)\sum_{s=1}^{c}(t_s - z_s)w_{sj}(z_s - z_s^2).
\end{aligned}
$$

Of course, the learning rule is then

$$
\begin{aligned}
\Delta w_{ji} &= -\eta\frac{\partial J}{\partial w_{ji}} \\
\Delta w_{kj} &= -\eta\frac{\partial J}{\partial w_{kj}},
\end{aligned}
$$

where the derivatives are as given above.

(b) We are given the cross-entropy criterion function

$$
J_{CE} = \sum_{k=1}^{c} t_k \ln\frac{t_k}{z_k}.
$$

The learning rule derivation is the same as in part (a) except that we need to replace $\partial J/\partial z$ with $\partial J_{ce}/\partial z$. Note that for the cross-entropy criterion we have $\partial J_{ce}/\partial z_k = -t_k/z_k$. Then, following the steps in part (a), we have

$$
\begin{aligned}
\frac{\partial J_{ce}}{\partial w_{kj}} &= y_j\sum_{s\neq k}^{c}\frac{t_k}{z_k}z_s z_k - y_j\frac{t_k}{z_k}(z_k - z_k^2) \\
&= y_j\sum_{s\neq k}^{c} t_k z_k - y_j t_k(1 - z_k).
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
\frac{\partial J_{ce}}{\partial w_{ji}} &= x_i f'(net_j)\sum_{s=1}^{c}\frac{t_s}{z_s}\sum_{r\neq x}^{c} w_{rj} z_s z_r \\
&\quad - x_i f'(net_j)\sum_{s=1}^{c}\frac{t_s}{z_s}w_{sj}(z_s - z_s^2).
\end{aligned}
$$

Thus we find

$$\frac{\partial J_{ce}}{\partial w_{ji}} = x_i f'(net_j) \sum_{s=1}^{c} t_s \sum_{r \neq s}^{c} w_{rj} z_r - x_i f'(net_j) \sum_{s=1}^{c} t_s w_{sj} (1 - z_s).$$

Of course, the learning rule is then

$$\Delta w_{ji} = -\eta \frac{\partial J_{ce}}{\partial w_{ji}}$$
$$\Delta w_{kj} = -\eta \frac{\partial J_{ce}}{\partial w_{kj}},$$

where the derivatives are as given above.

22. In the two-category case, if $g_1 \simeq P(\omega_1|x)$, then $1 - g_1 \simeq P(\omega_2|x)$, since we can assume the categories are mutually exclusive and exhaustive. But $1 - g_1$ can be computed by a network with input-to-hidden weights identical to those used to compute g_1. From Eq. 27 in the text, we know

$$\sum_{k_1} \int [g_{k_1}(\mathbf{x}, \mathbf{w}) - P(\omega_{k_1}|\mathbf{x})]^2 \, d\mathbf{x} + \sum_{k_2} \int [g_{k_2}(\mathbf{x}, \mathbf{w}) - P(\omega_{k_2}|\mathbf{x})] \, d\mathbf{x}$$

is a minimum. this implies that every term in the above equation is minimized.

Section 6.7

23. Consider the weight update rules given by Eqs. 12 and 23 in the text.

(a) The weight updates are have the factor $\eta f'(net)$. If we take the sigmoid $f_b(h) = \tanh(bh)$, we have

$$f_b'(h) = 2b \frac{e^{-bh}}{(1 + e^{-bh})^2} - 1 = 2b \underbrace{\frac{1}{e^{bh} + e^{-bh} + 2}}_{D} - 1 = 2bD = 1.$$

Clearly, $0 < D < 0.25$ for all b and h. If we assume D is constant, then clearly the product $\eta/\gamma f'_{\gamma b}(h)$ will be equal to $\eta f_b'(h)$, which tells us the increment in the weight values will be the same, preserving the convergence time. The assumption will be approximately true as long as $|bh|$ is very small or kept constant in spite of the change in b.

(b) If the input data is scaled by $1/\alpha$, then the increment of weights at each step will be exactly the same. That is,

$$\frac{\eta}{\gamma} f'_{\gamma b}(h/\gamma) = \eta f'(h).$$

Therefore the convergence time will be kept the same. (However, if the network is a multi-layer one, the input scaling should be applied to the hidden units' outputs as well.)