Appendix C

The multi-layer perceptron

The multi-layer perceptron (MLP) neural network shown in figure C.1 is the most typical of the models used in current neural networks research. It is the work-horse of many practical systems. The MLP architecture will be described here and its backpropagation training algorithm will be derived.

The network is made up of several layers of artificial neurons. Each neuron is connected to every neuron in the immediately adjacent layers. The following notation is used:

L	The number of layers.
N_ℓ	The number of neurons in layer ℓ.
P	The number of training patterns.
u^p_{1j}	The value of input neuron j (in layer 1) for pattern p ($j = 1 \cdots N_1$, $p = 1 \cdots P$)
$u^p_{\ell j}$	The output of neuron j in layer ℓ for pattern p ($j = 1 \cdots N_\ell$, $\ell = 1 \cdots L$).
$a^p_{\ell j}$	The "activation" of neuron j in layer ℓ for pattern p ($\ell = 2 \cdots L$).
$w_{\ell ji}$	The weight connecting neuron i in layer $\ell - 1$ to neuron j in layer ℓ.
$\theta_{\ell j}$	The "threshold" for neuron j in layer ℓ.
d^p_j	The desired output (the desired value of a^p_{Lj}) for training pattern p.
E_p	The sum squared error (at the outputs) for training pattern p.
E	The sum squared error over all training patterns.

The inputs are u^p_{1j} and the outputs are a^p_{Lj}. Each neuron of the MLP (in the hidden and output layers) does the following:

$$u^p_{\ell j} \;=\; f(a^p_{\ell j}) \qquad \text{where} \qquad a^p_{\ell j} = \theta_{\ell j} + \sum_{i=1}^{N_{\ell-1}} w_{\ell ji}\, u^p_{\ell-1,i} \qquad (\ell = 2 \cdots L) \tag{C.1}$$

The neuron transfer function is usually the sigmoid function:

$$f(x) \;=\; \frac{1}{1 + e^{-x}} \tag{C.2}$$

A useful fact about this function is that $f'(x) = f(x)(1 - f(x))$. The MLP is trained using *backpropagation*. The object of backpropagation is to adjust the weights so that, for all training patterns, the sum squared difference between the desired and actual output (the error) is minimized, i.e.:

$$\text{minimize:} \qquad E \;=\; \sum_{p=1}^{P} E_p \tag{C.3}$$

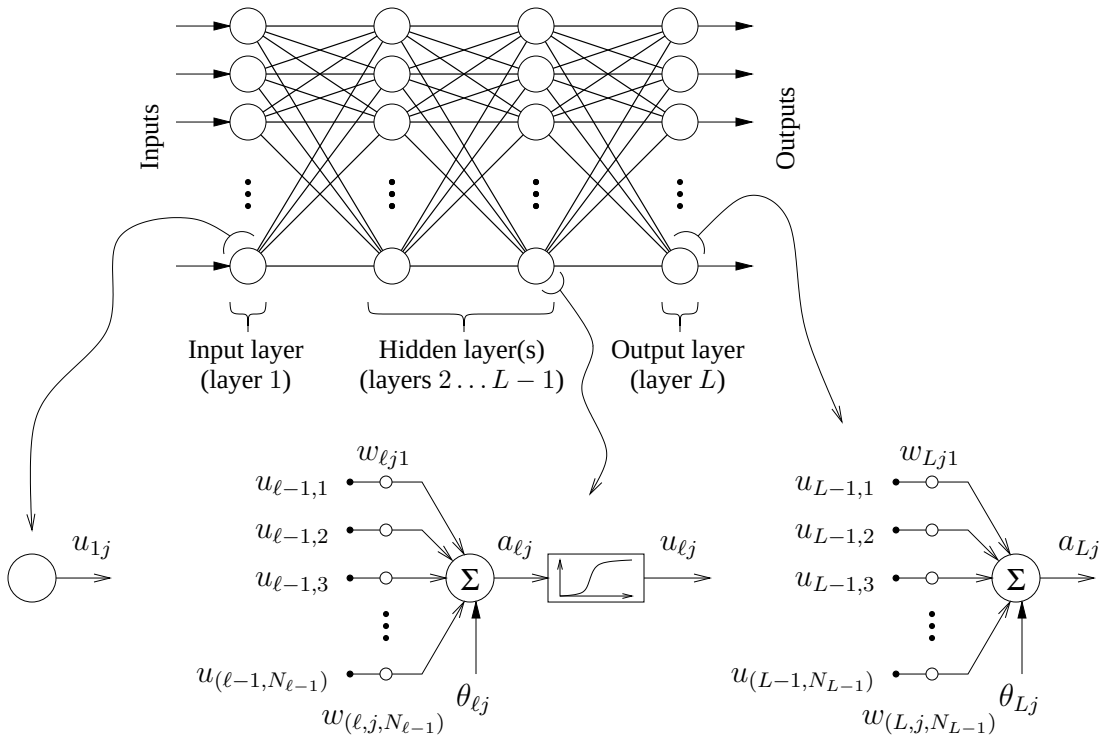

Figure C.1: *The multi-layer perceptron neural network.*

where

$$E_p = \sum_{i=1}^{N_L} \left(a_{Li}^p - d_i^p\right)^2 \tag{C.4}$$

The weights are adjusted by gradient descent:

$$w_{\ell ji} \leftarrow w_{\ell ji} - \alpha \frac{\partial E}{\partial w_{\ell ji}} \qquad (\alpha \text{ is the learning rate}) \tag{C.5}$$

$$\leftarrow w_{\ell ji} - \alpha \sum_{p=1}^{P} \frac{\partial E_p}{\partial w_{\ell ji}} \tag{C.6}$$

$$\leftarrow w_{\ell ji} - \alpha \begin{cases} \displaystyle\sum_{p=1}^{P} \frac{\partial E_p}{\partial u_{\ell j}^p} \cdot \frac{\partial u_{\ell j}^p}{\partial w_{\ell ji}} & (\text{if} \quad \ell = 2 \cdots (L-1)) \\ \displaystyle\sum_{p=1}^{P} \frac{\partial E_p}{\partial a_{Lj}^p} \cdot \frac{\partial a_{Lj}^p}{\partial w_{Lji}} & (\text{if} \quad \ell = L) \end{cases} \tag{C.7}$$

Now,

$$\frac{\partial u_{\ell j}^p}{\partial w_{\ell ji}} = \frac{\partial a_{\ell j}^p}{\partial w_{\ell ji}} f'(a_{\ell j}^p) \tag{C.8}$$

$$= u_{\ell-1,i}^p f'(a_{\ell j}^p) \tag{C.9}$$

$$= u_{\ell-1,i}^p u_{\ell j}^p (1 - u_{\ell j}^p) \tag{C.10}$$

Similarly,

$$\frac{\partial\, a^p_{Lj}}{\partial\, w_{Lji}} \;=\; u^p_{L-1,i} \tag{C.11}$$

Using the chain rule the $\partial E/\partial u$ of layer ℓ are associated with the $\partial E/\partial u$ of layer $\ell+1$:

$$D^p_{\ell j} = \frac{\partial\, E_p}{\partial\, u^p_{\ell j}} \;=\; \sum_{m=1}^{N_{\ell+1}} \frac{\partial\, E_p}{\partial\, u^p_{\ell+1,m}} \cdot \frac{\partial\, u^p_{\ell+1,m}}{\partial\, u^p_{\ell j}} \qquad \text{for } \ell = 1 \cdots (L-2) \tag{C.12}$$

$$= \sum_{m=1}^{N_{\ell+1}} \frac{\partial\, E_p}{\partial\, u^p_{\ell+1,m}} \cdot \frac{\partial\, a^p_{\ell+1,m}}{\partial\, u^p_{\ell j}} \cdot f'(a^p_{\ell+1,m}) \tag{C.13}$$

$$= \sum_{m=1}^{N_{\ell+1}} \frac{\partial\, E_p}{\partial\, u^p_{\ell+1,m}} \cdot w_{\ell+1,m,j} \cdot u^p_{\ell+1,m}(1 - u^p_{\ell+1,m}) \tag{C.14}$$

And similarly

$$\frac{\partial\, E_p}{\partial\, u^p_{L-1,j}} \;=\; \sum_{m=1}^{N_L} \frac{\partial\, E_p}{\partial\, a^p_{Lm}} \cdot w_{Lmj} \tag{C.15}$$

For the output layer the $\partial E/\partial a$ can be computed directly:

$$\frac{\partial\, E_p}{\partial\, a^p_{Lj}} \;=\; \frac{\partial \left[\sum_{i=1}^{N_L} \Big(a^p_{Li} - d^p_i\Big)^2\right]}{\partial\, a^p_{Lj}} \tag{C.16}$$

$$= 2\Big(a^p_{Lj} - d^p_j\Big) \tag{C.17}$$

The procedure for computing $\partial E/\partial\theta_{\ell j}$ is similar. The entire backpropagation algorithm is thus:

- Compute all $a^p_{\ell j}$ and $u^p_{\ell j}$ by forward propagation.
- For $p = 1 \cdots P$ and $j = 1 \cdots N_{L-1}$, compute

$$D^p_{L-1,j} = 2 \sum_{m=1}^{N_L} \Big(a^p_{Lm} - d^p_m\Big) w_{Lmj}$$

- For $p = 1 \cdots P$ and $\ell = (L-2) \cdots 2$ and $j = 1 \cdots N_\ell$, compute

$$D^p_{\ell j} = \sum_{m=1}^{N_{\ell+1}} D^p_{\ell+1,m} \cdot w_{\ell+1,m,j} \cdot u^p_{\ell+1,m}(1 - u^p_{\ell+1,m})$$

- For $j = 1 \cdots N_L$ and $i = 1 \cdots N_{L-1}$,

$$w_{Lji} \leftarrow w_{Lji} - 2\alpha \sum_{p=1}^{P} (a_{Lj} - d^p_j) u^p_{L-1,i}$$

- For $j = 1 \cdots N_L$,

$$\theta_{Lj} \leftarrow \theta_{Lj} - 2\alpha \sum_{p=1}^{P} \left(a_{Lj}^p - d_j^p \right)$$

- For $\ell = (L-1) \cdots 2$ and $j = 1 \cdots N_\ell$ and $i = 1 \cdots N_{\ell-1}$,

$$w_{\ell ji} \leftarrow w_{\ell ji} - \alpha \sum_{p=1}^{P} D_{\ell j}^p u_{\ell-1,i}^p u_{\ell j}^p (1 - u_{\ell j}^p)$$

- For $\ell = (L-1) \cdots 2$ and $j = 1 \cdots N_\ell$,

$$\theta_{\ell j} \leftarrow \theta_{\ell j} - \alpha \sum_{p=1}^{P} D_{\ell j}^p u_{\ell j}^p (1 - u_{\ell j}^p)$$

Backpropagation gets its name from the fact that the quantities $D_{\ell j}^p$ are computed by propagating their values backwards through the network.