The Backpropagation Algorithm

It will simplify our development of the backpropagation algorithm if we use the abbreviated notation for the multilayer network, which we introduced in Chapter 2. The three-layer network in abbreviated notation is shown in Figure 11.7.

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

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \quad \text{for } m = 0, 1, \ldots, M-1, \tag{11.6}$$

where M is the number of layers in the network. The neurons in the first layer receive external inputs:

$$\mathbf{a}^{0} = \mathbf{p}, \tag{11.7}$$

which provides the starting point for Eq. (11.6). The outputs of the neurons in the last layer are considered the network outputs:

$$\mathbf{a} = \mathbf{a}^{M}. \tag{11.8}$$

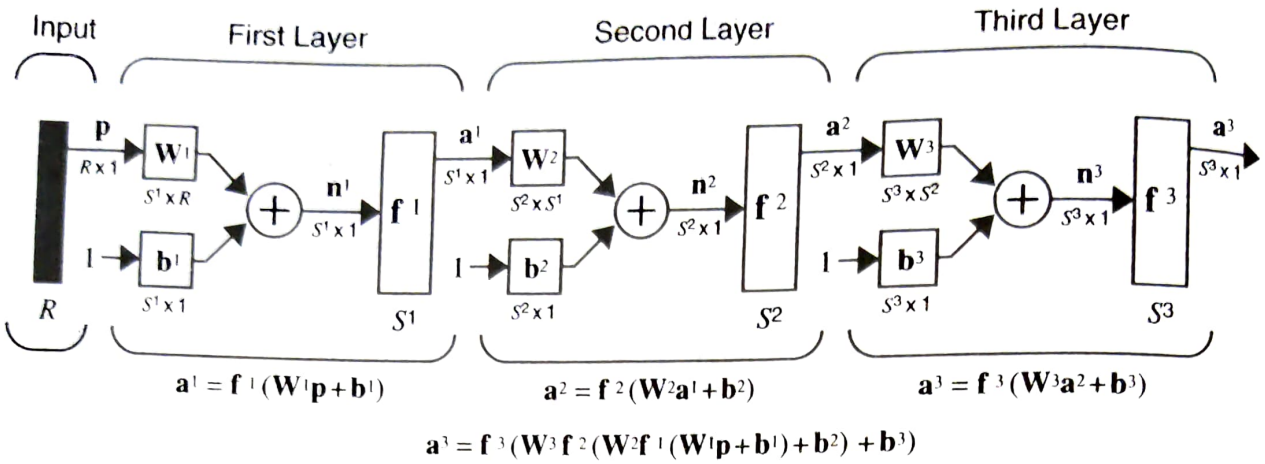


Figure 11.7 Three-Layer Network, Abbreviated Notation

Performance Index

The backpropagation algorithm for multilayer networks is a generalization of the LMS algorithm of Chapter 10, and both algorithms use the same performance index: *mean square error*. The algorithm is provided with a set of examples of proper network behavior:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}, \tag{11.9}$$

where $\mathbf{p}_q$ is an input to the network, and $\mathbf{t}_q$ is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The algorithm should adjust the network parameters in order to minimize the mean square error:

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] \tag{11.10}$$

where $\mathbf{x}$ is the vector of network weights and biases (as in Chapter 10). If the network has multiple outputs this generalizes to

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})] \tag{11.11}$$

As with the LMS algorithm, we will approximate the mean square error by

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k), \tag{11.12}$$

where the expectation of the squared error has been replaced by the squared error at iteration k

The steepest descent algorithm for the approximate mean square error is

$$w^m_{i,j}(k+1) = w^m_{i,j}(k) - \alpha \frac{\partial \hat{F}}{\partial w^m_{i,j}}, \tag{11.13}$$

$$b^m_i(k+1) = b^m_i(k) - \alpha \frac{\partial \hat{F}}{\partial b^m_i}, \tag{11.14}$$

where α is the learning rate.

So far, this development is identical to that for the LMS algorithm. Now we come to the difficult part – the computation of the partial derivatives.

Chain Rule

For a single-layer linear network (the ADALINE) these partial derivatives are conveniently computed using Eq. (10.33) and Eq. (10.34). For the multilayer network the error is not an explicit function of the weights in the hidden layers, therefore these derivatives are not computed so easily.

Because the error is an indirect function of the weights in the hidden layers, we will use the chain rule of calculus to calculate the derivatives. To review the chain rule, suppose that we have a function f that is an explicit function only of the variable n. We want to take the derivative of f with respect to a third variable w. The chain rule is then:

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} \tag{11.15}$$

For example, if

$$f(n) = e^n \text{ and } n = 2w, \text{ so that } f(n(w)) = e^{2w}, \tag{11.16}$$

then

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (e^n)(2) \tag{11.17}$$

We will use this concept to find the derivatives in Eq. (11.13) and Eq. (11.14):

$$\frac{\partial \hat{F}}{\partial w^m_{i,j}} = \frac{\partial \hat{F}}{\partial n^m_i} \times \frac{\partial n^m_i}{\partial w^m_{i,j}}, \tag{11.18}$$

$$\frac{\partial \hat{F}}{\partial b^m_i} = \frac{\partial \hat{F}}{\partial n^m_i} \times \frac{\partial n^m_i}{\partial b^m_i}. \tag{11.19}$$

The second term in each of these equations can be easily computed, since the net input to layer m is an explicit function of the weights and bias in that layer:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m . \tag{11.20}$$

Therefore

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \quad \frac{\partial n_i^m}{\partial b_i^m} = 1 . \tag{11.21}$$

If we now define

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}, \tag{11.22}$$

(the *sensitivity* of $\hat{F}$ to changes in the ith element of the net input at layer m), then Eq. (11.18) and Eq. (11.19) can be simplified to

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1}, \tag{11.23}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m . \tag{11.24}$$

We can now express the approximate steepest descent algorithm as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1}, \tag{11.25}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{11.26}$$

In matrix form this becomes:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T, \tag{11.27}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m, \tag{11.28}$$

where

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix} \tag{11.29}$$

(Note the close relationship between this algorithm and the LMS algorithm of Eq. (10.33) and Eq. (10.34)).

Backpropagating the Sensitivities

It now remains for us to compute the sensitivities $\mathbf{s}^m$, which requires another application of the chain rule. It is this process that gives us the term *backpropagation*, because it describes a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m+1$.

To derive the recurrence relationship for the sensitivities, we will use the following Jacobian matrix:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix} \tag{11.30}$$

Next we want to find an expression for this matrix. Consider the i,j element of the matrix:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1}\frac{\partial a_j^m}{\partial n_j^m} \tag{11.31}$$

$$= w_{i,j}^{m+1}\frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m(n_j^m) ,$$

where

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} . \tag{11.32}$$

Therefore the Jacobian matrix can be written

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1}\dot{\mathbf{F}}^m(\mathbf{n}^m) , \tag{11.33}$$

where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m(n_{S^m}^m) \end{bmatrix} \tag{11.34}$$

We can now write out the recurrence relation for the sensitivity by using the chain rule in matrix form:

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)\,(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} \tag{11.35}$$

$$= \dot{\mathbf{F}}^m(\mathbf{n}^m)\,(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

Now we can see where the backpropagation algorithm derives its name. The sensitivities are propagated backward through the network from the last layer to the first layer:

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \dots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1 . \tag{11.36}$$

At this point it is worth emphasizing that the backpropagation algorithm uses the same approximate steepest descent technique that we used in the LMS algorithm. The only complication is that in order to compute the gradient we need to first backpropagate the sensitivities. The beauty of backpropagation is that we have a very efficient implementation of the chain rule.

We still have one more step to make in order to complete the backpropagation algorithm. We need the starting point, $\mathbf{s}^M$, for the recurrence relation of Eq. (11.35). This is obtained at the final layer:

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M}. \quad (11.37)$$

Now, since

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M), \quad (11.38)$$

we can write

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M) \quad (11.39)$$

This can be expressed in matrix form as

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}) \quad (11.40)$$

Summary

Let's summarize the backpropagation algorithm. The first step is to propagate the input forward through the network:

$$\mathbf{a}^0 = \mathbf{p}, \quad (11.41)$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, \ldots, M-1, \quad (11.42)$$

$$\mathbf{a} = \mathbf{a}^M \quad (11.43)$$

The next step is to propagate the sensitivities backward through the network:

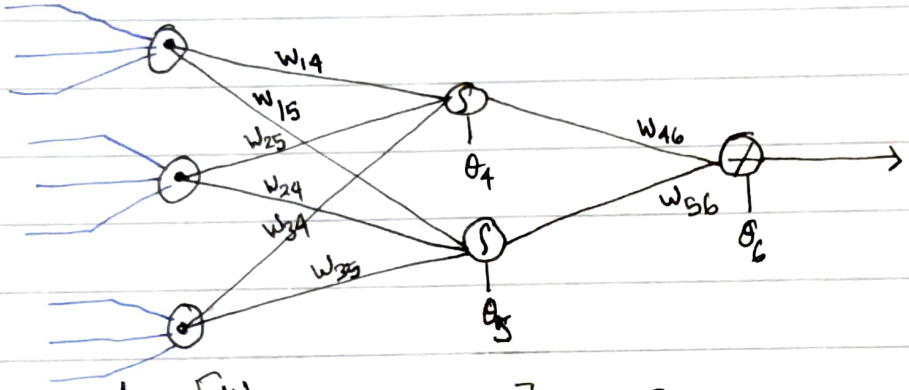
$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}), \quad (11.44)$$

Expt. No.

Page No.
Date.

2)a) Explain the Backpropagation learning algorithm.

b) Figure below shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in table 1, along with the first training tuple, $X = (1, 0, 1)$ whose class label is 1.



$$W^1 = \begin{bmatrix} w_{14} & w_{24} & w_{34} \\ w_{15} & w_{25} & w_{35} \end{bmatrix} \qquad W^2 = \begin{bmatrix} -0.3 & -0.2 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} -0.4 \\ -0.2 \end{bmatrix} \qquad b^2 = [0.1]$$

$$\alpha = 0.9 \qquad p = [1 \ 0 \ 1]$$

W^n = weight of a layer ✓

b^n = bias of a layer. ✓

α = learning rate. ✓

p = Input. ✓

Transfer function used in layer 1 = logsigmoid.

Transfer function used in layer 2 = Purelin.

OXFORD

Teacher's Signature .....

1st iteration.

$$a^0 = p$$

$$a^1 = f^1(w^1 p^T + b^1) = f^1\left[\begin{bmatrix} 0.2 & 0.4 & -0.5 \\ -0.3 & 0.1 & 0.2 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix}\right]$$

$$= f^1\left[\begin{bmatrix} -0.3 \\ -0.1 \end{bmatrix} + \begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix}\right] = f^1\left[\begin{bmatrix} -0.7 \\ 0.1 \end{bmatrix}\right] = \begin{bmatrix} \frac{1}{1+e^{-0.7}} \\ \frac{1}{1+e^{0.1}} \end{bmatrix}$$

$$a^1 = \begin{bmatrix} 0.668 \\ 0.475 \end{bmatrix}$$

$$a^2 = f^2(w^2 a_2 + b^2) = f^2\left[\begin{bmatrix} -0.3 & -0.2 \end{bmatrix}\begin{bmatrix} 0.668 \\ 0.475 \end{bmatrix} + [0.1]\right]$$

$$= f^2\left[[-0.2954] + [0.1]\right] = f^2[-0.1954] = [-0.1954]$$

Now, according to the consideration on transfer function for each layer, the derivative of each transfer function must be calculated as follows:

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

Log sigmoid:

$$= \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1-a^1)(a^1)$$

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

purelin:

Expt. No.

Page No.
Date.

So, the sensitivity is calculated as follows:

$$S^2 = -2\,F^2(n^2)\,(t - a^2)$$

$$= -2 \cdot (1) \cdot [1 + 0.1954) = -2.3908$$

$$S^1 = F^1(n^1)\,(W^2)^T S^2$$

$$= \begin{bmatrix} (1-a^1)(a^1) & 0 \\ 0 & (1-a^1)(a^1) \end{bmatrix} (W^2)^T S^2$$

$$= \begin{bmatrix} (1-0.668)(0.668) & 0 \\ 0 & (1-0.475)(0.475) \end{bmatrix} \begin{bmatrix} -0.3 \\ -0.2 \end{bmatrix} [-2.390$$

$$= \begin{bmatrix} 0.222 & 0 \\ 0 & 0.249375 \end{bmatrix} \begin{bmatrix} -0.3 \\ -0.2 \end{bmatrix} (-2.3908)$$

$$S^1 = \begin{bmatrix} -0.0944 \\ -0.0499 \end{bmatrix}$$

According to the backpropagation formula the weight is calculated as follows:-

$$W^2 = W^2 - \alpha S^2 [a^1]^T$$

$$= [-0.3 \quad -0.2] - (0.9)(-2.3908)([0.668 \; 0.475$$

$$= [1.138 \quad 0.82]$$

OXFORD

Teacher's Signature .....

$$W^1 = W^1 - \alpha S^1 (a^0)^T$$

$$= \begin{bmatrix} 0.2 & 0.4 & -0.5 \\ -0.3 & 0.1 & 0.2 \end{bmatrix} - (0.9) \begin{bmatrix} -0.0444 \\ -0.0499 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2 & 0.4 & -0.5 \\ 0.3 & 0.1 & 0.2 \end{bmatrix} - (0.9) \times \begin{bmatrix} -0.0444 & 0 & -0.0444 \\ -0.0499 & 0 & -0.0499 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2 & 0.4 & -0.5 \\ 0.3 & 0.1 & 0.2 \end{bmatrix} - \begin{bmatrix} -0.03996 & 0.0 & -0.03996 \\ -0.04491 & 0.0 & -0.04491 \end{bmatrix}$$

$$W^1 = \begin{bmatrix} 0.23996 & 0.4 & -0.46004 \\ -0.25509 & 0.1 & 0.24491 \end{bmatrix}$$

The bias of each layer is calculated as follows:

$$b^2 = b^2 - \alpha S^2$$

$$= [0.1] - (0.9)(-2.3908)$$

$$= 2252$$

$$b^1 = b^1 - \alpha S^1$$

$$= \begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix} - (0.9) \begin{bmatrix} -0.0444 \\ -0.0499 \end{bmatrix}$$

$$= \begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix} - (0.9) \begin{bmatrix} -0.03996 \\ -0.04491 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} -0.36 \\ 0.245 \end{bmatrix}$$

Expt. No.

Page No.
Date.

2/a) Back-propagation learning Algorithm is described as follows:—

Step 1: The first step is to propagate the input forward through the network:

$$\text{where } a^{m+1} = f^{m+1}\left(W^{m+1} a^m + b^{m+1}\right)$$

for $m = 0, 1, \ldots M-1$

$$\text{So, } a = a^M.$$

Step 2: Calculation of sensitivity is required and it is the process which give us the term back-propagation. The sensitivity describes a recurrance relationship in which sensitivity of the layer m is calculated from the sensitivity at layer $m+1$.

The sensitivity of layer can be calculated is as follows:

$$S^m = F^m(n^m)\left(W^{m+1}\right)^T S^{m+1}$$

where,

$$F^m(n^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & 0 \\ 0 & \dot{f}^m(n_2^m) & 0 \\ 0 & 0 & \dot{f}(n_{s^m}^m) \end{bmatrix}$$

W^{m+1} = weight of $m+1$ layer

S^{m+1} = sensitivity of $m+1$ layer.

OXFORD

Teacher's Signature .....

For the final layer the sensitivity at final layer is calculated as follows:—

$$S_i^M = -2 F^M(n^M)(t-a)$$

$$F^M(n^M) = \begin{bmatrix} f^m(n_1^m) & 0 & 0 \\ 0 & f^m(n_2^m) & 0 \\ 0 & 0 & f^m(n_3^m) \end{bmatrix}$$

t = target of each ~~layer~~ input.

a = output of final layer.

Step3: The weight and biases are updated using steepest descent rule:

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

and $$b^m(k+1) = b^m(k) - \alpha s^m$$

Step4: The convergence coitercia for backpropagation algorithm is sum of squred error (SSE). If it is considered that the performance index is $F(x)$ then it is calculated as follows:

$$F(x) = \sum_{i=1}^{N} e^T e = \sum_{i=1}^{N} (t-a)^T (t-a)$$

The network parameter is adjusted in order to minimize the above squred error. ~~else~~ and if not minimized then go to step 1.

For the final layer the sensitivity at final layer is calculated as follows:—

$$S_i^M = -2 F^M(n^M)(t-a)$$

$$F^M(n^M) = \begin{bmatrix} f^m(n_1^m) & 0 & 0 \\ 0 & f^m(n_2^m) & 0 \\ 0 & 0 & f^m(n_3^m) \end{bmatrix}$$

t = target of each ~~layer~~ input.

a = output of final layer.

Step3: The weight and biases are updated using steepest descent rule:

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

and $$b^m(k+1) = b^m(k) - \alpha s^m$$

Step4: The convergence coiteria for backpropagation algorithm is sum of squred error (SSE). If it is considered that the performance index is $F(x)$ then it is calculated as follows:

$$F(x) = \sum_{i=1}^{N} e^T e = \sum_{i=1}^{N} (t-a)^T (t-a)$$

The network parameter is adjusted in order to minimize the above squred error ~~else~~ and if not minimized then go to step 1.