APPENDIX – A

BACK-PROPAGATION ALGORITHM

Back-propagation algorithm works on gradient descent method. For each training pattern presented to the input layer of the network, error at the nodes in the output layer of the network is calculated. By using the error obtained, the weights and thresholds between layers are updated in order to minimize the error of the network.

Let,

\( y_j \) be the inner product of the weights with the outputs of the nodes in the previous layer. It is given by:

\[
y_j = \sum w_{ij} x_i + \theta_i \quad (A1)
\]

\( x_i \) the output of the non-linear function \( f() \), and it is given by:

\[
X_i = \frac{1}{1+\exp(\sum w_{ij} x_i + \theta_i)} \quad (A2)
\]

Fig A-1 Forward calculation in a network
E(p) the error of a pattern, and it is given by:

$$E(p) = \frac{1}{2} \sum (d(p) - x(p))^2$$  \hspace{1cm} (A3)

P  pattern number

\( \eta \)  learning factor

\( \Delta w_{ij} \)  the amount of weight to be added or subtracted with the previous weight, and

\( \delta \)  the error at each node in the hidden layers and in the output layer.

Back-propagation algorithm refers to the propagation of error of the nodes from the output layer to the nodes in the hidden layers. These errors are used to update the weights of the network. The amount of weights to be added or subtracted to the previous weight is governed by delta rule.

The delta rule is given by:

$$\Delta w_{ij} \propto - \frac{\partial E(p)}{\partial w_{ij}} = \eta \delta_i x_i$$  \hspace{1cm} (A4)

Since change in the error is a function of change in the input to a node, and change in the input to a node is a function of change in the weight, a chain rule is used to obtain the following expression.

$$\frac{\partial E(p)}{\partial w_{ij}} = \frac{\partial E(p)}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}}$$  \hspace{1cm} (A5)

where

$$\frac{\partial y_i}{\partial w_{ij}} = x_i$$  \hspace{1cm} obtained from equation A1
where

$$\frac{\partial x_i}{\partial y_j} = f'(y_j)$$

For the nodes in the output layer

$$\frac{\partial E(p)}{\partial x_i} = -(d_i - x_i)$$ obtained from equation A3 \hfill (A7)

For the nodes in the hidden layers

$$\frac{\partial E(p)}{\partial x_i} = \sum \frac{\partial E(p)}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$ \hfill (A8)

where

$$\frac{\partial y_j}{\partial x_i} = w_{ij}$$ obtained from equation A1

By substituting equation A7 in equation A7 in A6,

$$\delta_i (output\ layer) = f'(y_i) (d_i - x_i)$$ \hfill (A9)

where

$$f'(y_i) = x_i (1-x_i)$$

By substituting equation A8 in equation A6,

$$\delta_i (Hidden\ layer) = f'(y_i) \sum \delta_j w_{ij}$$ \hfill (A10)

Step 1: Initialize the weights and thresholds randomly between layers.

Step 2: Present the inputs of a pattern and compute the outputs by equation A2.
Step 3: Calculate the error of a pattern by equation A3

Step 4: Compute $\delta$ in the output layer by equation A6.

Step 5: Update the weights between layers and thresholds in the layers by:

$$W_{ij}(p+1) - w_{ij}(p) + \eta \delta_j x_i \quad (A11)$$

$$\theta_i(p+1) - \theta_i(p) + \eta \delta_i \quad (A12)$$

Step 6: Compute $\delta$ for the nodes in the hidden layers by equation A10 and update weights by equation A11 and thresholds by equation A12.

Step 7: Repeat from step 2 to step 6 for the remaining training patterns and check if the performance of the network is reached. If so, stop training the network; else repeat from step 2 to step 7.

For a flow-chart of the BPA algorithm, please refer to Figure 2.2.