Chapter 6

Modified Backpropagation Algorithm for Single Hidden Layer Feedforward Neural Networks

6.1 Introduction

Abid et al. (2001) have described a modified standard backpropagation algorithm (MBP) using the sum of the squares of the linear and nonlinear errors of all output units and for the current pattern. Asari (2001) has shown that a multilayer feedforward multiple valued neural network can be trained using a multilevel nonlinear threshold function with backpropagation learning rule. Ngom et al. (2000) have proposed permutably homogeneous perceptron learning algorithm using multiple valued multiple threshold functions. Yamamoto and Nikiforuk (2000) have proposed new algorithms using a non gradient but algebraic method. Yu et al. (2002) have developed a general backpropagation learning algorithm for FNN
with time varying non gradient but algebraic method.

Ergezinger and Thomson (1995) have presented a new learning procedure which is based on a linearization of the nonlinear processing elements and the optimization of the multilayer perceptron layer by layer. Wang and Chen (1996) have proposed a faster new learning algorithm based on layer by layer optimization procedure. The dynamic forgetting factors method makes their proposed new algorithm more powerful in dynamic system identification. Kwok and Yeung (1997) have studied different objective functions for training new hidden units in constructive neural networks. They have followed layer by layer optimization technique. Yam and Chow (1997) have proposed extended least squares based algorithm for training feedforward networks. In this, the weights connecting the last hidden and output layers are first evaluated by least squares algorithm. The weights between input and hidden layers are then evaluated using a modified gradient descent algorithm. This layer by layer optimization eliminates the stalling problem experienced by the pure least squares type algorithm. Mc Loone et al. (1998) have presented a new hybrid optimization strategy for training feedforward neural networks. This algorithm combines gradient based optimization of nonlinear weights with singular value decomposition computation of linear weights in one integrated routine. They have shown that this method is superior to second order method. The natural gradient method has an ideal dynamic behaviour which resolves the slow learning speed of the standard gradient descent method caused by plateaus. Rubanov (2000) has proposed an efficient learning method for a feedforward network that combines the backpropagation strategy and optimization layer by layer.
Layerwise optimization is constructed using the Taylor series expansion of non-linear operators describing a feedforward network and proposed to update weights of each layer by the backpropagation based Kaczmarz iterative procedure.

In this chapter, a modified form of the backpropagation based fast training algorithm for single hidden layer FNN is proposed. In this algorithm, hidden and output layers are trained separately. The output layer is trained using a modified standard backpropagation algorithm proposed by Abid et al. (2001). Fictitious teacher signals for the output of hidden layer neurons are determined algebraically as described by Yamamoto and Nikiforuk (2000). Then hidden layer is trained using a proposed optimization criterion. Network is trained until convergence is reached. The efficiency in terms of time and iteration of the proposed method is shown by the simulation results of the circle in the square, 4-byte parity checker, L-T character recognition, and function approximation problems.

6.2 Training of single hidden layer neural network

Single hidden layer neural network is considered. The activation function of hidden layer neurons are assumed to be sigmoidal. The output layer neuron(s) can have linear or non-linear activation function depending on the problem. In this chapter, networks with single output neuron having linear activation function is considered. The input to the neural network may be received from outside the network or from the output layer.
6.2.1 Training method

In the input layer bias node is included with the value 0.5. The linear and nonlinear output of the $j^{th}$ neuron in the $l^{th}$ layer for the current pattern are respectively as follows:

$$nc\ell_j^l = \sum_{i=1}^{n} w_{ji}^l y_{i}^{l-1}$$  \hspace{1cm} (6.1)

$$f(nc\ell_j^l) = \frac{1}{1 + e^{-nc\ell_j^l}} = y_j^l$$  \hspace{1cm} (6.2)

where $n$ represents number of neurons in the layer $l - 1$. The linear and nonlinear errors are respectively as follows:

$$e_{1,j}^l = f^{-1}(d_j^l) - nc\ell_j^l$$  \hspace{1cm} (6.3)

$$e_{2,j}^l = d_j^l - y_j^l$$  \hspace{1cm} (6.4)

Let us consider the following optimization criterion, which was defined by Abid et al. (2001), to train the output layer neurons:

$$E_p = \frac{1}{2} \lambda (e_{1,j}^o)^2 + \frac{1}{2} (e_{2,j}^o)^2$$  \hspace{1cm} (6.5)

where $p$ represents the current pattern, $j$ represents $j^{th}$ neuron, $o$ represents output layer and $\lambda$ represents a weighting coefficient. The following weight update rule derived by Abid et al. (2001) is used here to update the output layer weights:

$$\Delta w_{ji}^l = \mu \lambda e_{1,j}^l y_{i}^{l-1} + \mu f'(nc\ell_j^l) e_{2,j}^l y_{i}^{l-1}$$  \hspace{1cm} (6.6)

where $\mu$ is the learning coefficient. Fictitious teacher signals are needed for the output of hidden layer neurons. Yamamoto and Nikiforuk (2000) have determined the temporal teacher signal for the output of hidden layer neurons, which is used
to find the linear error of the output of hidden neurons. Temporal teacher signal has been calculated as follows:

\[ \Delta y^h = W^\dagger v_i^o \]  \hspace{1cm} (6.7)

\[ y^h_i = H(y^h + \Delta y^h) \]  \hspace{1cm} (6.8)

\( y^h, \Delta y^h, v_i^o, y^h_i \) are vectors, \( y_i \) is a temporal teacher signal, \( h \) represents hidden layer, \( W^\dagger \) is a weight matrix and \( H \) represents the operator. \( W^\dagger \) is defined as follows:

\[ W^\dagger = \begin{cases} W(W^TW)^{-1} & \text{if the number of columns of } W \geq \text{the number of rows of } W \\ (WW^T)^{-1}W & \text{if the number of rows of } W \geq \text{the number of columns of } W \end{cases} \] \hspace{1cm} (6.9)

where \( W \) is the weight matrix connecting the hidden layer neurons and output layer neurons. The operator \( H \) is used by multiplying an appropriate scalar for a vector \((y^h + \Delta y^h)\) if the range space includes the origin as an inner point, or deleting an extra amount of each element of it. Using the temporal teacher signal the linear error \( e^h_{3j} \) of the \( j^{th} \) neuron of the hidden layer may be calculated as follows:

\[ e^h_{3j} = f^{-1}(y^h_{ij}) - net^h_j \] \hspace{1cm} (6.10)

Let us define the new optimization criterion of the hidden layer for the \( j^{th} \) neuron of the current pattern \( p \) as follows:

\[ E_{tp} = \frac{1}{2} \lambda (e^o_{1j})^2 + \frac{1}{2} (e^o_{2j})^2 + \frac{1}{2} \gamma (e^h_{3j})^2 \] \hspace{1cm} (6.11)

where \( \gamma \) represents another weighting coefficient. The necessity of the weighting coefficient was discussed in Abid et al. (2001). Now the weight update rule for
hidden layer may be derived as follows:

\[
\Delta w_{ij}^h = -\mu \frac{\partial E_1}{\partial w_{ij}^h} \\
= \mu \lambda \sum_{j=1}^{n} \frac{\partial w^0_j}{\partial y^0_i} \frac{\partial y^0_i}{\partial w_{ij}^h} \frac{\partial w^h_i}{\partial w_{ij}^h} + \mu \sum_{j=1}^{n} \frac{\partial y^0_j}{\partial y^0_i} \frac{\partial w^0_j}{\partial y^0_i} \frac{\partial y^0_i}{\partial w_{ij}^h} \frac{\partial w^h_i}{\partial w_{ij}^h} + \mu \gamma \epsilon^0_i \frac{\partial w^0_i}{\partial w_{ij}^h} (6.12)
\]

\[
\Delta w_{ij}^h = \mu \lambda y_{ij}^{-1} c_1 + \mu y_{ij}^{-1} f'(u_{ij}^h) c_2 + \mu \gamma \epsilon^0_i y_{ij}^{-1} (6.13)
\]

where \( u = \text{net} \), \( h = 1 \) represents input layer, \( n \) represents the number of neurons in the output layer,

\[
c_1 = f'(u_{ij}^h) \sum_{j=1}^{n} \epsilon^0_j w_{ij}^0 (6.14)
\]

\[
c_2 = \sum_{j=1}^{n} \epsilon^0_j f'(u_{ij}^h) w_{ij}^0 (6.15)
\]

and weights will be updated by

\[
w_{ji} = w_{ji} + \Delta w_{ji} (6.16)
\]

In the proposed scheme, we first find the output of the FNN for the selected input. Then train the output layer for selected number of times using (6.6) and (6.16). After that the hidden layer is trained for a predefined number of times using (6.13) and (6.16). The same weight updating procedure is carried out for all input patterns. This training process is repeated for the selected network structure until the desired accuracy is obtained.

### 6.2.2 Algorithm

1. Initialize the weights of the selected network structure using
uniformly distributed random numbers.

step 2. Perform step 3 and then 4 for each input pattern.

step 3. Train the output layer.

Find the output of the FNN.

Calculate the weights to be added for the output layer using (6.6), and update the weights of the output layer using (6.16).

Repeat step 3 for a predefined number of times.

step 4. Train the hidden layer.

Find output for the FNN.

Find temporal teacher signal for the output of the hidden layer neurons using (6.7), (6.8) and (6.9).

Calculate the weights to be added for the hidden layer using (6.13).

Update the weights of the hidden layer using (6.16).

Repeat step 4 for a predefined number of times.

step 5. Repeat steps 2 to 4 until the desired accuracy is obtained.

6.2.3 Complexity of the algorithm

The worst case time complexity of the proposed algorithm is obtained in terms of the number of iterations performed. Using layer by layer optimization every
input pattern is trained. For a single pattern, weight updation is carried out for predefined number of times say $M$. If number of input patterns available is $N$ then for all pattern of one cycle it requires $N$ iterations for each layer. If the training needs $P$ cycles to get the required accuracy for the desired value of the output neurons, it requires totally $M \times N \times P$ iterations for the output layer. As the hidden layer optimization require one more iteration to obtain (6.13) than (6.16) because of (6.7), (6.8) and (6.9), to be calculated for each hidden neuron and for each input pattern, the iterations required for hidden layer weight updation is $(1 + k) \times M \times N \times P$, where $k$ represents number of hidden neurons. Therefore, the total computational effort of the proposed algorithm is $M \times N \times P + (1 + k) \times M \times N \times P$.

6.3 Simulation results

The performance of the new algorithm is compared with the Abid et al. (2001) modified standard backpropagation algorithm and our earlier work described in chapter 5. In the proposed algorithm, one iteration means the execution of instructions from step 2 to step 4 for all input patterns once. In simulation, for each input pattern, input and output layers are separately trained 10 number of times for all applications. The algorithm is used to train the network with 25 different initial weights, for all the selected applications.

6.3.1 Circle in the square problem

In this problem, we have considered a circle of radius 0.35. The neural network has to decide whether a point of coordinate $(x,y)$ varying from -0.5 to +0.5 lies
inside the circle or outside. The desired output selected to represent a position to be inside circle is 0.1 and outside is 0.9. 100 different input patterns were used to train the network. Termination condition used in the training was 0.01. Sigmoidal activation function is used in hidden and output layers. The learning rate $\mu=0.008$, the weighting coefficients $\lambda=4.0$, $\gamma=0.0005$ and 4 hidden neurons are used in the training. It has been found that the network converged with an average of 16.6 iterations and 0.679 seconds whereas MBP (Abid et al., 2001) required in average 167 epochs and 10.49 seconds to converge to the mean squared error (MSE) of 0.1 with 8 hidden neurons, $\mu=0.5$ and $\lambda=0.01$. Figure 6.1 shows the decrease in error corresponding to the iterations of the network. Table 6.1 shows the average convergence time, the iterations required and the learning parameters used for this application.

<table>
<thead>
<tr>
<th>Method</th>
<th># Hidden Neurons</th>
<th>Time(s) for Convergence</th>
<th>Iteration</th>
<th>Termination Condition MSE</th>
<th>$\mu$</th>
<th>$\lambda$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBP</td>
<td>8</td>
<td>2.49</td>
<td>208.58</td>
<td>0.01</td>
<td>0.5</td>
<td>0.7</td>
<td>--</td>
</tr>
<tr>
<td>Proposed</td>
<td>4</td>
<td>0.679</td>
<td>44.76</td>
<td>0.01</td>
<td>0.008</td>
<td>4.0</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison table for the circle in the square problem

6.3.2 4-byte parity checker

This application determines the parity of a 4-byte binary number. In training
Figure 6.1: Learning curve based on MSE and number of iterations of the circle in the square problem.

-0.5 and 0.5 were used to represent 0 and 1 respectively for the input patterns. Desired outputs are taken to be 0.1 and 0.9 to represent the lower and higher levels respectively. Network weights are initialized by the range of values between -5 and +5. Termination condition of MSE = 0.0001, is used in training. Sigmoidal activation function is used for both the hidden and output layer. MBP (Abid et al., 2001) required in average 82 iterations and 0.06 seconds for the convergence of 8 hidden neurons FNN with the learning rate $\mu=10.0$ and weighting coefficient $\lambda=0.01$. The proposed algorithm in average required 15.15 iterations and 0.07 seconds for the convergence with 7 hidden neurons FNN and with the parameters $\mu=0.04$, $\lambda=54.0$, and $\gamma=0.005$. Figure 6.2 shows the learning curve for the proposed algorithm. Table 6.2 shows the convergence time and number of iterations and hidden neurons required for this application.
<table>
<thead>
<tr>
<th>Method</th>
<th># Hidden Neurons</th>
<th>Time(s) for Convergence</th>
<th>Iteration</th>
<th>Termination Condition MSE</th>
<th>$\mu$</th>
<th>$\lambda$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBP</td>
<td>8</td>
<td>0.06</td>
<td>82</td>
<td>0.0001</td>
<td>10.0</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>7</td>
<td>0.07</td>
<td>15.15</td>
<td>0.0001</td>
<td>0.04</td>
<td>54</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

Table 6.2: Comparison table for the parity checker problem

Figure 6.2: Learning curve based on MSE and number of iterations of the parity checker.
6.3.3 L-T character recognition

In the letters L and T character recognition problem, the training set consisted of a 3 x 3 pixel binary image for each letter with its four orientations, in total eight patterns. The letters L and T were indicated by the target values 0.1 and 0.9 for the outputs and -0.5 and 0.5 for the inputs. For this application, the proposed algorithm required network with only one hidden and one output neuron. Network weights are initialized with the range of values -1.5 and 1.5. Sigmoidal activation is used for the output layer neuron and the bipolar activation function for the hidden layer neuron. The learning rate $\mu=0.5$, the weighting coefficients $\lambda=0.6$ and $\gamma=0.005$ are used in training the network. In simulation it has been found that this algorithm required in average 3.1 iterations and 0.012 seconds to converge to the termination condition of $\text{SSE} = 0.0001$ whereas our earlier algorithm presented in chapter 5 required in average 350 iterations and 3 seconds to converge to the same termination condition with the same FNN. Abid et al. (2001) algorithm converged in average 0.05 seconds and 21 iterations with the parameter values $\mu=0.007$ and $\lambda=23.0$. Table 6.3 shows the comparative study of the proposed algorithm with our algorithm described in chapter 5 and Abid et al. MBP algorithm.

6.3.4 Function approximation problem

We consider the following function for approximation,

$$z(x, y) = \frac{\sin[0.5(x + 5)^2 + (y - 5)^2]}{(x + 5)^2 + (y - 5)^2} + \frac{\sin[0.5(x - 5)^2 + (y - 3)^2]}{(x - 5)^2 + (y - 3)^2},$$

where $(x, y) \in [-20, 20]$. In this problem, 250 sets of input patterns are trained on a
<table>
<thead>
<tr>
<th>Method</th>
<th># Hidden Neurons</th>
<th>Time(s) for Convergence</th>
<th>Iteration</th>
<th>Termination Condition MSE</th>
<th>$\mu$</th>
<th>$\lambda$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBP</td>
<td>1</td>
<td>0.06</td>
<td>31</td>
<td>0.0001</td>
<td>0.007</td>
<td>23.0</td>
<td>-</td>
</tr>
<tr>
<td>Previous Work (Chapter 5)</td>
<td>1</td>
<td>3.0</td>
<td>350</td>
<td>0.0001</td>
<td>0.05</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>Proposed</td>
<td>1</td>
<td>0.04</td>
<td>4.1</td>
<td>0.0001</td>
<td>0.5</td>
<td>0.6</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison table for the L-T character recognition problem

network 2-5-1 with logistic neurons in the hidden layer and a linear neuron in the output layer. Network is trained with 15 different initial weights. In average, 57 epochs are needed to reach the MSE of 0.002454566 within 2 seconds. The trained network is tested with 2500 set of points generated from a regularly spaced grid on $[-20,20]^2$. The MSE of the tested set is 0.005822. Lera and Pintzas (2002) have used 2-50-1 network to approximate this function for which the MSE greater than $10^{-3}$.

6.4 Conclusion

An algorithm with different optimization criterions to minimize linear and nonlinear error of the output and hidden layer neurons is proposed. Fictitious teacher signals for the hidden layer outputs are determined to train the hidden layer weights. Training of hidden layer and output layer was considered separately.
for each input pattern in the proposed algorithm to speed up the convergence.

It has been found that the time and iterations needed for circle in the square problem and 4-byte parity problem to converge to the termination condition is better than Abid et al. (2001) MBP algorithm. L-T character recognition problem needs minimum number of epochs and time to reach the termination condition than Abid et al. MBP algorithm and our algorithm described in chapter 5.

The robust learning of the proposed algorithm has been shown. The effectiveness of the proposed algorithm is shown by the tables in terms of the time, iteration and hidden nodes needed for the convergence. The initialization of the network plays an important role in convergence. Finding the optimal value for learning rate and weighting coefficient was very difficult task in this procedure.