Chapter 7

Genetic Algorithm based Supervised Learning for Multilayer Feedforward Network

7.1 Introduction

In the previous chapter, a GA based weight determination algorithm for Backpropagation Networks is discussed. The same technique could be extended for application in Multilayer Feedforward Neural Networks (MFNN). MFNNs, which adopt supervised learning procedures, have their error surface highly convoluted with hills and valleys in multi-dimensional space. Also, the long learning phase (number of iterations) owing to slow convergence, results in a substantial use of computing time and power, if not resulting in instability.

However, GAs on the other hand have turned out to be robust search and optimization techniques outperforming gradient based techniques in obtaining solutions to problems acceptably well, fairly accurately and above all, acceptably quickly.

In this chapter, such a GA based supervised learning procedure for MFNN is discussed. The conventional supervised learning for MFNN is briefly reviewed before discussing the GA based learning algorithm. The performance of conventional MFNN has been compared with the GA based MFNN. The problem of buckling of non-prismatic continuous beams discussed in the earlier chapter has been chosen as a test suite problem to study the inferring capability of the two networks.
Neural Networks have established themselves as viable alternatives to statistical techniques in modeling the performance of the system. In this chapter, the behavior of GA based MFNN as a non linear regression correlator is also discussed.

7.2 Learning in Multilayer Feedforward Neural Networks

An MFNN typically consists of an input layer, an output layer and one or more intermediate layers termed hidden layers. The neurons in the input layer are connected to those in the hidden layer which in turn are connected to the succeeding layer viz., the next hidden layer or the output layer, as the case may be (Refer Fig. 1.4).

The activity of input neurons in the input layer represents the raw information that is fed into the network. The activity of hidden layer neurons is determined by the activities of the neurons in the preceding layer (input layer/hidden layers) and the connecting weights between the neurons of the two layers. Similarly, the activity of the neurons in the output layer depends on the activity of the neurons in the preceding layer and the connecting weights.

The synaptic weight vector \( \mathbf{W} \) represents the accumulated knowledge and the input vector \( \mathbf{X} \) a new information. MFNN learns new input patterns by changing the strength of the synaptic weights appropriately, thereby reducing the distance between the accumulated knowledge and the new input information.

Most of the NN models undergo a learning process during which the synaptic weights are adjusted. Learning algorithms have been broadly categorized as supervised and unsupervised algorithms. Supervised learning
algorithms employ an external reference signal and generate an error signal by comparing the reference with the obtained response. In contrast, unsupervised learning schemes do not incorporate a reference signal and are dependent on principles of self organization. The work discussed here confines itself to supervised learning. In supervised learning, based on the error signal, the synaptic weights are adjusted to improve its learning ability. The schema for the supervised learning algorithm has been presented in Fig. 7.1 [87].

A general expression for the supervised learning algorithm is

\[ W_{i}^{(t+1)} = W_{i}^{(t)} + \Delta W_{i}^{(t+1)} \]  

(7.1)

where,

\[ \Delta W_{i}^{(t+1)} = \eta x_{i}^{(t+1)} (y_{T}^{(t+1)} - y^{(t+1)}) + \alpha \Delta W_{i}^{(t)} \]  

(7.2)

and \( W_{i}^{(t)} \) is the weight corresponding to the input \( x_{i}^{(t)} \). The term \( \Delta W_{i}^{(t)} \) is the change in weight, \( \eta \) the learning rate and \( \alpha \) is the momentum factor. \( y_{T}^{(t)} \) is the target (desired) output and \( y^{(t)} \) is the calculated (actual) response of the neuron. The proper selection of \( \eta \) to a chosen problem is of critical importance. While a small value of \( \eta \) may result in slow learning, a large value of \( \eta \) though resulting in faster learning, may result in rendering the system unstable. For instance, Backpropagation learning algorithm, which is one of the most popular supervised learning algorithms, encounters the problem of local minimum. Also, the number of learning steps is often so high that the learning phase has intensive calculations.
7.3 GA based MFNN

Assume an MFNN whose network configuration is $l - m_1, m_2 - n$ (1 input neurons, $m_1, m_2$ neurons in the first and second hidden layers respectively, and $n$ output neurons). The number of weights that are to be determined are $N_W = (l + m_2)m_1 + m_2.n$. With each weight being a real number and assuming the number of digits (string length) in the weight to be $d$, a string $S$ of decimal values representing the $N_W$ weights and therefore having a string length $L = N_W.d$, is randomly generated. The string $S$ represents the weight matrices of the input-hidden and hidden-output layers, in a linear form, arranged according to row major or column major order as selected by the designer.

Let $x_1, x_2, \ldots, x_d, \ldots, x_L$ represent a chromosome and $x_{kd+1}, x_{kd+2}, \ldots, x_{(k+1)d}$ represent the $k^{th}$ gene in the chromosome. The actual weight $w_k$ is given by

$$w_k = \begin{cases} 
+ \frac{(x_{kd+2}10^{d-1} + x_{kd+3}10^{d-2} \ldots x_{(k+1)d})}{10^{d-1}}, & \text{if } 5 \leq x_{kd+1} \leq 9 \\
- \frac{(x_{kd+2}10^{d-1} + x_{kd+3}10^{d-2} \ldots x_{(k+1)d})}{10^{d-1}}, & \text{if } 0 \leq x_{kd+1} < 5 
\end{cases}$$

(7.3)

The initial population comprises of such randomly generated chromosomes.

**Fitness function:** The fitness function must be devised for each problem to be solved. The following algorithm illustrates the procedure:
**ALGORITHM MFNN_FITGEN()**

{ 
\[ (\vec{I}_i, \vec{T}_i), i = 1, 2, ..., N \] where \( \vec{I}_i = (I_{1i}, I_{2i}, ..., I_{li}) \) and 
\( \vec{T}_i = (T_{1i}, T_{2i}, ..., T_{ni}) \) represent the input output pairs of the problem to be solved by MFNN with a configuration \( l - m_1, m_2, ..., m_k - n. \)

For each chromosome \( C_i, i = 1, 2, ..., p \) belonging to the current population \( P_i \) whose size is \( p \)

\{ 
Extract weights \( W_i \) from \( C_i \) with the help of result (7.3);
Keeping \( W_i \) as a fixed weight setting, train the MFNN for the \( N \) input instances;
Calculate error \( E_i \) for each of the input instance using the formula,
\[
E_i = \sum_j (T_{ji} - O_{ji})^2 
\] 
(7.4)
where \( \overline{O}_i \) is the output vector calculated by MFNN;
Find the root mean square \( E \) of the errors \( E_i, i = 1, 2, ..., N \) 
\[
i.e. \quad E = \sqrt{\frac{\sum_i E_i}{N}} 
\] 
(7.5)
Calculate the fitness value \( F_i \) for each of the individual string of the population as 
\[
F_i = \frac{1}{E} 
\] 
(7.6)
\}

Output \( F_i \) for each \( C_i, i = 1, 2, ..., p \).
\}

END MFNN_FITGEN.

Reproduction: In this phase, the parents are selected in pairs at random. The chromosomes of the respective pairs are recombined using a standard Two point Cross Over operator. The offspring, which now form the current population again, have their fitness calculated as illustrated by the algorithm MFNN_FITGEN().
Convergence: A population is said to have converged when 95% of the individuals constituting the population share the same fitness value.

In summary, the algorithm for the GA based supervised learning is as follows:

\begin{verbatim}
ALGORITHM GA_NN_SL()
{
    i ← 0 ;
    Generate the initial population \( P_i \) of real coded chromosomes \( C_j^i \) each representing a weight set for the MFNN;

    While the current population \( P_i \) has not converged
    {
        Generate fitness values \( F_j^i \) for each \( C_j^i \in P_i \) using the algorithm MFNN_FITGEN();

        Get the mating pool ready by exchanging worst fit individuals and duplicating high fit individuals;

        Using the Cross Over mechanism reproduce offspring from the parent chromosomes;

        \( i \leftarrow i + 1 ; \)

        Call the current population \( P_i \);
        Calculate fitness values \( F_j^i \) for each \( C_j^i \in P_i \);
    }

    Extract weights from \( P_i \) to be used by the MFNN;
}
END GA_NN_SL.
\end{verbatim}
7.4 Performance Comparison

Experiment I: Error-Iteration relation

In this phase, a uniform configuration $l - m - n$ is chosen for both the networks, viz., MFNN and GA based network. A sample set $I = \{(x_i, y_i)\}$ of inputs representing a well-defined problem is presented as a training set to the networks. The error-iteration curve (Fig. 7.2) for both networks turned out to be monotonically converging. However, as a significant observation GA based networks converged amazingly quickly, when compared to that of the conventional MFNNs.

Experiment II: Inferring Capability

In this phase, the inferring capability of the networks for both the trained and untrained (unseen) inputs are observed. The problem of Buckling of non prismatic thin walled beams is chosen as a test suite. For the non prismatic continuous beam shown in Fig. 6.7a the buckling load found out using the interaction curves plotted as shown in Fig. 6.7b, is used as a training set for the GA based network. The sample results are given in Table 7.1. The fitness-generation history curves for different depth taper $\alpha$ are shown in Fig. 7.3.

7.5 GA based MFNN as Non linear Regression Correlator

Investigations [58, 116] have observed the similarity of the behavior of NN models to that of Statistical models. However, for most problem domains, especially those that are described by Non linear models, NN
systems have either displayed better predictions or helped to predict solutions when their statistical counterparts could not.

In this section, the behavior of the GA based MFNN as a correlator of data points having a non-linear relationship is presented.

Several methods of statistical inference exist which help predict the relationship between a population of data points by making use of a sample of them. Least squares regression is an established technique that helps prediction if the inferred relationship between the data sets is linear or could be mapped to a linear system. However, in the case of nonlinear models, statistical methods present special difficulties. The difficulty is compounded when the techniques adopted attempt to locate the lowest point (global minimum) of a multi dimensional surface. If the surface has multiple valleys (that is, it is not unimodal), most numerical techniques can do no better than to find a local minimum (one of the valleys) rather than the global minimum.

It is in such a scenario that NN models have a role to play. A test suite problem - viz., a closed convex region (sphere), represented by the equation $x^2 + y^2 + z^2 = c^2$ is chosen for a case study. The data points are restricted to the positive quadrant of the sphere. The objective behind choosing such a system is to select a model which conventional statistical regression would not be able to handle.

On the other hand, GA-MFNN is trained with a sample of data points ($D_s$) and the inferring capability tested over (i) the sample $D_s$ for which the network is trained (Trained instances) and (ii) a sample for which the network is not trained (Untrained instances). The results are presented in Table 7.2.
The observation is, when the data points clustered around the periphery (the surface of the sphere in this case) the model behaved at its best. The performance quite naturally, deteriorated as the points tended to scatter and drift away from the periphery.

7.6 Summary

In this chapter, a GA based supervised learning rule for Multilayer Feed forward Networks has been discussed. When compared with conventional MFNN, the GA based model has been observed to display an acceptably good performance. The behavior of the network has been tested over a problem chosen from Structural Engineering viz., buckling of non prismatic continuous beams. Finally, the behavior of GA based MFNN as a non linear regression correlator has been discussed.
Table 7.1
Sample results for Non prismatic continuous beam

<table>
<thead>
<tr>
<th>Inference</th>
<th>Data Set</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>Exact</th>
<th>Calculated by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trained</td>
<td>0.2119</td>
<td>0.8220</td>
<td>0.1695</td>
<td>0.169270</td>
</tr>
<tr>
<td></td>
<td>Untrained</td>
<td>0.0848</td>
<td>0.8051</td>
<td>0.1695</td>
<td>0.171432</td>
</tr>
<tr>
<td>Trained</td>
<td></td>
<td>0.3898</td>
<td>0.7627</td>
<td></td>
<td>0.168323</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5932</td>
<td>0.5763</td>
<td></td>
<td>0.168850</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7288</td>
<td>0.3559</td>
<td></td>
<td>0.169455</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8305</td>
<td>0</td>
<td></td>
<td>0.169887</td>
</tr>
<tr>
<td>Untrained</td>
<td></td>
<td>0.2373</td>
<td>0.8051</td>
<td>0.1683</td>
<td>0.169314</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.4237</td>
<td>0.7373</td>
<td></td>
<td>0.168374</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6356</td>
<td>0.5508</td>
<td></td>
<td>0.168603</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8136</td>
<td>0.6780</td>
<td></td>
<td>0.165346</td>
</tr>
<tr>
<td>Trained</td>
<td></td>
<td>0.8305</td>
<td>0.9237</td>
<td>0.4237</td>
<td>0.427166</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8983</td>
<td>0.4492</td>
<td></td>
<td>0.428235</td>
</tr>
<tr>
<td></td>
<td>Untrained</td>
<td>0.4237</td>
<td>0.8898</td>
<td>0.4237</td>
<td>0.423573</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8051</td>
<td>0.6780</td>
<td></td>
<td>0.423650</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8898</td>
<td>0.5085</td>
<td>0.4237</td>
<td>0.425116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9068</td>
<td>0.3390</td>
<td></td>
<td>0.425116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8983</td>
<td>0.1695</td>
<td></td>
<td>0.425317</td>
</tr>
<tr>
<td>Trained</td>
<td></td>
<td>0.4237</td>
<td>0.8898</td>
<td>0.4237</td>
<td>0.427486</td>
</tr>
<tr>
<td></td>
<td>Untrained</td>
<td>0.8051</td>
<td>0.6780</td>
<td></td>
<td>0.428296</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8898</td>
<td>0.5085</td>
<td>0.4237</td>
<td>0.427726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9068</td>
<td>0.3390</td>
<td></td>
<td>0.427726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8983</td>
<td>0.1695</td>
<td></td>
<td>0.425317</td>
</tr>
<tr>
<td>Trained</td>
<td></td>
<td>0.1695</td>
<td>0.9746</td>
<td>0.6780</td>
<td>0.677608</td>
</tr>
<tr>
<td></td>
<td>Untrained</td>
<td>0.0847</td>
<td>0.9492</td>
<td>0.6780</td>
<td>0.678142</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1271</td>
<td>0.9492</td>
<td></td>
<td>0.678060</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2797</td>
<td>0.9746</td>
<td></td>
<td>0.677802</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9746</td>
<td>0.1695</td>
<td></td>
<td>0.677894</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9407</td>
<td>0.0170</td>
<td></td>
<td>0.678139</td>
</tr>
</tbody>
</table>
Table 7.2
Rejection Analysis

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Inference Set</th>
<th>Rejection</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Closed Convex region - Positive Octant of a Sphere)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Points on the surface ( {(x, y, z): x^2 + y^2 + z^2 = 1} ) ( x, y, z &gt; 0 )</td>
<td></td>
<td>15.38</td>
<td>40.0</td>
</tr>
<tr>
<td>Points on and above the surface ( {(x, y, z): x^2 + y^2 + z^2 \geq 1} ) ( x, y, z &gt; 0 )</td>
<td></td>
<td>41.18</td>
<td>57.14</td>
</tr>
<tr>
<td>Points on and within region ( {(x, y, z): x^2 + y^2 + z^2 \leq 1} ) ( x, y, z &gt; 0 )</td>
<td></td>
<td>72.2</td>
<td>71.43</td>
</tr>
<tr>
<td>Positive infinite space ( {(x, y, z): x^2 + y^2 + z^2 &lt; 1} ) ( x, y, z &gt; 0 )</td>
<td></td>
<td>72.73</td>
<td>88.89</td>
</tr>
</tbody>
</table>
Fig. 7.1: Schema for the Supervised learning algorithm
Fig. 7.2: Error - iteration curve
Fig. 7.3: Fitness-generation history curve