CHAPTER 7

DEVELOPMENT OF ALGORITHM FOR FEATURE BASED PREDICTION WITH NEURAL NETWORK

7.1 INTRODUCTION

ARIMA model could be used for forecasting in non-stationary time series when linearity between the variables is valid. If the linearity is not valid, the ARIMA models do not produce effective results, and cause the forecast errors to increase. ANN is the next option for the prediction of future values, because of their ability to approximate any measurable function to any decision degree. This chapter gives the necessary descriptions in the development of prediction model with ANN, and with the extracted features of continuous glucose monitoring sensor data time series.

7.2 FEATURE BASED PREDICTION

The classical back propagation technique in NN follows the gradient of error surface to fix the weights of neurons. This gradient is an extremely local pointer and does not point to the global minimum. The classical back propagation is sensitive to parameters like learning rate and momentum factor. If the learning rate is too small, it would lead to slow convergence and a too large value would make the search direction jump wildly and never converge. The optimal values are difficult to be found out and often obtained empirically. Hence, the proposed work has availed a Feed Forward Neural Network (FFNN), trained with features extracted from the past values, for forecasting the BG values ahead in time. In the current work,
two custom-made transfer functions were used, one at the hidden layer neurons and other at the output layer.

The recurrent characteristics of BG dynamics could be tracked with a moving window of length ‘n’ with a step size of ‘p’. The incoming time series was represented in terms of features. Every member of the feature set contributed in training the neural network, by adjusting the learning rate parameter with weighted values of the extracted features. For every window, the features were computed and passed to the network for learning. The flow of works in FNN has been shown in Figure 7.1.

![Feature Based Neural Network](image)

**Figure 7.1  Steps for Feature Based Neural Network**

### 7.2.1  Customized Activation Functions

Mean ($M_k$) or average is a measure of central tendency and provides an idea of the concentration of the observations about the central part of the distribution. The variance ($V_k$), the second moment is a measure of how far a set of data is spread out from the mean. Skewness ($S_k$), the third moment gives the idea about the shape of the distribution. Positive or negative values of skewness furnish an idea whether the data are in increasing trend or in
decreasing trend and the amount of deviation from the central mean. Skew and its coefficient help to adjust for the deviation from mean. The height and sharpness of the peak relative to the rest of the data are measured by kurtosis ($K_k$). ApEn ($A_k$) is complementary to peak detection and is useful in evaluating both dominant and subordinate patterns in concentration time series. The features set could be represented as,

$$R_k = \{M_k, V_k, S_k, K_k, A_k\} \quad (7.1)$$

This feature set tracks the underlying variation of a dynamic signal. This helps to identify the principal components of the signal of interest. The feature set information is used in the learning rate estimation and firing of the activation functions in the hidden and output layers. As per the requirement of BP training process, the activation functions are bounded and differentiable. The learning rate parameter $\tilde{\alpha}_k$ is made adaptive by the linear combination of time varying features of BG dynamics. At each iteration, $\tilde{\alpha}_k$ value is tuned with $\{M_k, V_k, S_k, K_k, A_k\}$. This adaptiveness has overcome the drawbacks of classical BP algorithms i.e slow convergence due to small step size and diversification due to larger step size. This multi objective model is a more diversified model, reducing the risk and has performed will out of samples relative to traditional CGM models, as could be realized from the discussion that follows. In order to measure the diversification, approximate entropy, a widely accepted measure of diversity is used. It is known that greater the value of the entropy measure for CGM system, higher the CGM diversification is. The learning rate parameter $\tilde{\alpha}_k$ for $k$’th iteration could be represented as,

$$\tilde{\alpha}_k = C_{1,k} M_k + C_{2,k} V_k + C_{3,k} S_k + C_{4,k} K_k + C_{5,k} A_k \quad (7.2)$$

where, the coefficients $C_{i,k}$ are estimated through Least Squares (LS). The method of least squares is a standard approach to the approximate solution
of over determined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made on the results of every single equation (Ljung 1997). The expression for parameter estimation by Least Squares is given by,

$$\hat{\beta} = (X^T X)^{-1}X^T Y$$  \hspace{1cm} (7.3)$$

where, $\hat{\beta}$ is the set of parameters, $X$ is the array of inputs and $Y$ is the set of received data.

Since the proposed work has to work with non binary values, the usual sigmoidal function would not be effective. The next option is to go for exponential activation functions. The custom made non linear activation functions tuned with adaptive learning rate make more effectiveness than the traditional functions in terms of training performance and validation. The activation functions ($f_o$ and $f_h$) have been coined through a non linear equation and are based on the input data, their corresponding weights and the learning rate. The activation functions have been framed with an intention to have non linear characteristics in the hidden layer and output layer neurons' responses. The input is multiplied with its square root value first. To reduce larger deviation of input, the square root of input is subtracted from the product as a correctness factor. The resultant is multiplied with the learning rate $\tilde{\alpha}_k$ which accounts for the adaptive nature of the activation functions as per the input dynamics.

The Custom Activation function $f_h$ for hidden layer could be defined as follows,

$$f_h(x) = (x\sqrt{x} - \sqrt{x}) * \tilde{\alpha}_k$$  \hspace{1cm} (7.4)$$

Where $x$ is the sum of products of inputs (CGM data) and the corresponding weights applied to hidden layer nodes and $\tilde{\alpha}_k$ is the learning rate extracted
from the input window, applied in the activation of neurons. The CGM data obtained every five minutes is fed into the neural network. The activation function at output layer $f_o$ is given by

$$
f_o(p) = \left(1 + \sqrt{p}\right) \cdot \sqrt{p} \cdot \tilde{a}_k
$$

(7.5)

Where $p$ is the input to the output layer neurons which is equal to the sum of products of hidden layer outputs and the corresponding weights. The activation functions were used to transform the activation level of a node (neuron) into an output signal. The hidden units of neural network needed an activation function to introduce nonlinearity into the networks, which made the multilayer network more effective. The activation functions at the hidden and output layers were able to achieve their target with the assistance of adaptive learning rate parameter, which has been tuned by the features of incoming signal. Since the learning rule has a direct impact on the convergence of learning algorithms, suitable changes were made for new activation functions. Various possible transfer functions have been tried in this forecasting of glucose concentration with CGM time series. The above specified custom functions met the performance goal of minimum MSE and less computational time. This feature set acted as a multi objective model and made the learning rate parameter of the neural network adaptive.

### 7.3 PROPOSED ALGORITHM FOR FEATURE BASED PREDICTION

The FNN algorithm is formulated with the long-established feed forward back propagation procedure. However it has been customized with adaptive learning rate and specially designed activation functions at the hidden layer and output layer. The FNN algorithm has been applied with a
moving window concept to the CGM data set. The algorithm could be explained as follows.

**Step 1:** Define Window of size ‘n’.
- For each iteration, the window moves from 1 to n, 2 to n+1, 3 to n+2 and so on upto N-n where N is the length of the data set comprising of CGM data of 5 minutes sampling frequency.

**Step 2:** From each window, extract the feature set

\[ R_k = \{M_k, V_k, S_k, K_k, A_k\} \]

**Step 3:** Compute the Learning Rate,

\[ \tilde{\alpha}_k = C_{1,k} M_k + C_{2,k} V_k + C_{3,k} S_k + C_{4,k} K_k + C_{5,k} A_k \]

Where, the coefficients \( C_{i,k}\) are estimated with LS method.

**Step 4:** Compute the Layers’ responses,

- **Hidden Layer**, \( Y_{j}^{k} = f_{h} \left( \sum_{i} W_{ij} Z_{i}^{k} \right) \); \hspace{1cm} (7.6)

  - for \( j = 1,2,\ldots J \); for \( k = 1,2,\ldots K \);
  - \( K \) is the total number of training pairs, \( J \) is the number of neurons in the hidden layer.
  - Where \( Y_{j}^{k} \) is the output of \( j \)’th neuron of hidden layer at \( k \)’th iteration.
  - \( \sum_{i} W_{ij} Z_{i}^{k} \) is the product of inputs \( (Z_{i}^{k}) \) with their corresponding weights \( W_{ij} \) between input layer and hidden layer.
  - The activation function at hidden layer :

\[ f_{h}(x) = \left( x \sqrt{x} - \sqrt{x} \right) \ast \tilde{\alpha}_k \]

\[ \text{where } \chi = \sum_{i} W_{ij} Z_{i}^{k} \]

- **Output Layer**, \( O^{k} = f_{o} \left( \sum_{j} V_{j} Y_{j}^{k} \right) ; \hspace{1cm} (7.7) \)
Where \( Y^h_j \) is the output of \( j \)th neuron in the hidden layer.

\( V_j \) is the weight between the \( j \)th hidden layer neuron and output node.

The output layer activation function is

\[
f_o(p) = (1 + \sqrt{p}) * \sqrt{p} * \tilde{a}_k
\]

Where \( p \) is the input to output layer neurons which is equal to the sum of products of hidden layer outputs and the corresponding weights.

**Step 5:** Compute the error:

\[
E \leftarrow \left( \frac{1}{2} \right) ||d_k - O_k||^2
\]  

(7.8)

Where, \( d_k \) is the desired output and \( O_k \) is the actual output.

**Step 6:** Calculate error signal vectors, \( \delta_0 \) and \( \delta_y \).

\[
\delta_o = \left( \frac{1}{2} \right) \left[ (d_k - O_k) * \tilde{a}_k \right]
\]  

(7.9)

Where \( \delta_o \) is the error signal vector from output layer.

\[
\delta_y = W^t_j * \delta_0 * f^t_y
\]  

(7.10)

Where \( \delta_y \) is the error signal vector from hidden layer.

And \( f^t_y = \left( \frac{1}{2} \right) [1 - y^2_j] \)

(7.11)

**Step 7:** Adjust weights of the Output layer.

\[
V_{k+1} \leftarrow V_k + \tilde{a}_k * \delta_o * y^t
\]  

(7.12)

**Step 8:** Adjust weights of the Hidden layer.

\[
W_{k+1} \leftarrow W_k + \tilde{a}_k * \delta_y * z^t
\]  

(7.13)
- Repeat steps 2 to 7 for all the training patterns.
- If the training patterns are over for the training set,
  - Check for $E < E_{\text{max}}$?
    - If yes, stop training.
    - If not, assign $E \leftarrow 0$ and begin a new training cycle.

The training of neural network model involved the modification of the weight vector gradually, in order to minimize the difference between the predicted value and the desired response. Here, RMSE between the predicted and actual values has been considered as the performance measure.

### 7.4 IMPLEMENTATION OF FNN

The development of a simple feed forward back propagation neural network with specialized transfer functions at hidden and output layer are presented in this section. The functions were designed to get activated, according to the learning parameter ‘\( \alpha \)’, which was derived from the features of the input data. Here, the feature extraction has been used as the guidance in tracking the dynamics of nonlinear input. Thus, the neurons were made to generate the signal based on the derived feature, and not on instantaneous values.

In the proposed work, window sizes have been tried with $N = 4$ to $N=10$ and finalized with $N = 6$ through the false nearest neighbor method (Rhodes et al 1997). And this window size also goes in agreement with the half an hour data which consists of 6 samples taken once in five minutes. For each iteration of the training process, the moments viz, mean, variance, skewness, kurtosis and ApEn were computed in each window as shown in Figure 7.2.
Let the input to NN be:

At time \( t \)

\[
\begin{array}{cccccc}
X_{t-5} & X_{t-4} & X_{t-3} & X_{t-2} & X_{t-1} & X_t \\
\end{array}
\]

\[
\begin{array}{cccccc}
X_{t-4} & X_{t-3} & X_{t-2} & X_{t-1} & X_t & X_{t+1} \\
\end{array}
\]

\[
\begin{array}{cccccc}
X_{t+n-4} & X_{t+n-3} & X_{t+n-3} & X_{t+n-2} & X_{t+n-1} & X_{t+n} \\
\end{array}
\]

\[
\begin{array}{cccccc}
X_{t-5} & X_{t-4} & X_{t-3} & X_{t-2} & X_{t-1} & X_t \\
\end{array}
\]

\[
\begin{array}{cccccc}
X_{t-4} & X_{t-3} & X_{t-2} & X_{t-1} & X_t & X_{t+1} \\
\end{array}
\]

\[
\begin{array}{cccccc}
X_{t+n-4} & X_{t+n-3} & X_{t+n-3} & X_{t+n-2} & X_{t+n-1} & X_{t+n} \\
\end{array}
\]

**Figure 7.2 Moving window**

Mean, Variance, Skew and Kurtosis are obtained by the standard calculations explained in Chapter 2. ApEn is a mathematical algorithm created to measure the repeatability or predictability within a time series. ApEn is extremely sensitive to their input parameters: \( m \) (length of the data segment being compared), \( r \) (similarity criterion), and \( N \) (length of data).

There is no established consensus on parameter selection in short data sets, especially for biological data. Hence in this research, for ApEn(\( N,m,r \)), the length of time series, \( N \) was the length of window i.e., 6. For pattern length ‘\( m \)’, which is an integer that represents the length of compared run of data, it has been fixed with heuristic approach that it should be less than or equal to half the length of input sequence. The trials were made with \( m = 2 \) and \( m = 3 \). From the experiments it was observed that the trial with \( m \) equal to 2 produced improved results. Hence, the size of \( m \) has been fixed as 2. The similarity criterion, \( r \) which is a positive real number that specifies a filtering
level. The literature often uses \( m = 1 \) or \( 2 \) or \( 3 \), together with \( r \) proportional to the SD of data in the window. For the CGM sensor data sets used in current research, the \( r \) value varies in the range of 0.561 to 19.83. Presence of noise or any spurious signals, leads to higher \( r \) values.

The architecture of NN used for feature based prediction has been shown in Figure 7.3. It consists of an input layer and a hidden layer each consisting of six nodes, and an output layer with single neural node. The number of nodes in the input layer and hidden layer were finalized with false nearest neighborhood method. The activations functions used at the hidden and output layer were given in Equations (7.4) and (7.5) respectively.

![Figure 7.3 Architecture of FNN](image)

Based on the dynamics of BG values in the data window, the values of the features would be varying with the corresponding discrepancy in the coefficient values. The proposed method has been tested with the three data
sets. Each data set in turn were assigned for training, validation and testing based on the dividerand function (with values of target Data Size, trainRatio, valRatio, testRatio) which is the default function of neural networks. Here the data set size is 1440 and in that 60% of data were used for training, 20% for validation and 20% for testing.

The weights are initialized unequally in random. The weights between input layer neurons and hidden layer neurons, \( W_{i,j} \), where \( i \) equal to 6 and \( j \) equal to 6 were given in Equation (7.14). The weights between hidden layer neurons and output layer neurons, \( V_{i,k} \), where \( i \) equal to 6 and \( k \) equal to 1 were given in Equation (7.15).

\[
W_{i,j} = \begin{bmatrix}
0.2 & 0.4 & 0.8 & 1.6 & 3.2 & 6.4 \\
0.4 & 0.8 & 1.6 & 3.2 & 6.4 & 0.2 \\
0.2 & 0.4 & 0.8 & 1.6 & 3.2 & 6.4 \\
0.8 & 0.2 & 0.4 & 0.8 & 1.6 & 3.2 \\
1.6 & 0.8 & 0.2 & 0.4 & 0.8 & 1.6 \\
3.2 & 1.6 & 0.8 & 0.2 & 0.4 & 0.8
\end{bmatrix}
\]  

\[
V_{i,j} = \begin{bmatrix}
1 \\
0.8 \\
0.6 \\
0.4 \\
0.2 \\
0.1
\end{bmatrix}
\]  

(7.14)  

(7.15)

The sample of extracted features from each type of data set has been given in Table 7.1. These feature values in turn would be used for the computation of adaptive learning rate.
The adaptive learning rate parameter was formed by the sum of products of features and their corresponding coefficients \( C_{i,k} \). A sample of \( C_{i,k} \), coefficients / parameters of features obtained with the Least Squares (LS) method is given below in Equations (7.16) and (7.17).

\[
C_{1,k} = -0.0006, \quad C_{2,k} = 0.0005, \quad C_{3,k} = -0.0633, \quad C_{4,k} = 0.0725,
\]

\[
C_{5,k} = 0.2486 \quad ; \quad \bar{\alpha}_k = 0.1 \quad \ldots \quad (7.16)
\]

\[
C_{1,k} = -0.0010, \quad C_{2,k} = 0.0006, \quad C_{3,k} = -0.0718, \quad C_{4,k} = 0.1003,
\]

\[
C_{5,k} = 0.4196 \quad ; \quad \bar{\alpha}_k = 0.13 \quad \ldots \quad (7.17)
\]

<table>
<thead>
<tr>
<th>Sample Features from data sets</th>
<th>Mean</th>
<th>Variance</th>
<th>Skewness</th>
<th>Kurtosis</th>
<th>Approximate Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Set 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>104.355978</td>
<td>12.423090</td>
<td>-0.8138666</td>
<td>2.248130</td>
<td>0.223143</td>
<td></td>
</tr>
<tr>
<td>106.4010908</td>
<td>0.95599343</td>
<td>-0.37210727</td>
<td>1.562178</td>
<td>0.1133284</td>
<td></td>
</tr>
<tr>
<td>105.7021605</td>
<td>5.39023190</td>
<td>1.59259854</td>
<td>3.8424870</td>
<td>0.0408219</td>
<td></td>
</tr>
<tr>
<td>131.1475380</td>
<td>25.042124</td>
<td>0.05229354</td>
<td>1.703568</td>
<td>0.2231430</td>
<td></td>
</tr>
<tr>
<td>161.8862148</td>
<td>26.6778827</td>
<td>-0.9762888</td>
<td>2.537538</td>
<td>0.2231435</td>
<td></td>
</tr>
<tr>
<td><strong>Data Set 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77.8015866</td>
<td>0.6133966</td>
<td>0.0377160</td>
<td>1.7328977</td>
<td>0.1839228</td>
<td></td>
</tr>
<tr>
<td>75.8083633</td>
<td>0.5038059</td>
<td>0.0360257</td>
<td>1.7327735</td>
<td>0.1839228</td>
<td></td>
</tr>
<tr>
<td>73.9979516</td>
<td>0.4175535</td>
<td>0.0343361</td>
<td>1.7326640</td>
<td>0.1839228</td>
<td></td>
</tr>
<tr>
<td>72.3461700</td>
<td>0.3491634</td>
<td>0.0326937</td>
<td>1.7325592</td>
<td>0.1839228</td>
<td></td>
</tr>
<tr>
<td>70.8325150</td>
<td>0.2944798</td>
<td>0.0311420</td>
<td>1.7324624</td>
<td>0.1839228</td>
<td></td>
</tr>
<tr>
<td><strong>Data Set 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>151.6157033</td>
<td>2.5960465</td>
<td>-0.1584276</td>
<td>1.7483112</td>
<td>0.2231435</td>
<td></td>
</tr>
<tr>
<td>153.1884166</td>
<td>1.7902852</td>
<td>-0.1911095</td>
<td>1.7565596</td>
<td>0.2231435</td>
<td></td>
</tr>
<tr>
<td>154.4688966</td>
<td>1.1356176</td>
<td>-0.2376483</td>
<td>1.7708484</td>
<td>0.1133286</td>
<td></td>
</tr>
<tr>
<td>155.4591966</td>
<td>0.6352847</td>
<td>-0.3109354</td>
<td>1.7995064</td>
<td>0.1278333</td>
<td></td>
</tr>
<tr>
<td>156.1640166</td>
<td>0.2864235</td>
<td>-0.4455797</td>
<td>1.8724067</td>
<td>0.0408219</td>
<td></td>
</tr>
</tbody>
</table>
The extracted features from a set sample data sets, along with the computed learning rate and the prediction result in terms of RMSE in mg/dL were given in Table 7.2.

### Table 7.2 Sample performance of FNN model

<table>
<thead>
<tr>
<th>Subject ID</th>
<th>Sample Features</th>
<th>Adaptive Lr.Rate</th>
<th>RMSE in mg/dL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M_k$</td>
<td>$V_k$</td>
<td>$S_h$</td>
</tr>
<tr>
<td>#101</td>
<td>131.147</td>
<td>25.042</td>
<td>0.05229</td>
</tr>
<tr>
<td>#138</td>
<td>73.997</td>
<td>0.4175</td>
<td>0.03433</td>
</tr>
<tr>
<td>#151</td>
<td>155.459</td>
<td>0.6352</td>
<td>-0.3109</td>
</tr>
<tr>
<td>#017</td>
<td>106.401</td>
<td>0.9559</td>
<td>-0.3721</td>
</tr>
<tr>
<td>#062</td>
<td>75.8081</td>
<td>0.5038</td>
<td>0.0360</td>
</tr>
</tbody>
</table>

The method proposed was tested with different scenarios such as, training and testing of network with same patient data, training with one patient data and testing with another patient data. This assisted to check the ability of the network to account for inter individual variability of SNR of the CGM sensor data. The variations in blood glucose signal due to physiological and behavioral changes would be reflected in the extracted features which in turn assist in tracking the intra individual variability of SNR by tuning of network parameters accordingly.

Feature set was extracted from each window and used for learning rate adjustment and activation functions at the hidden layer and the output layer, which were applied at back propagation of error during training process. As given in the algorithm, final weights of hidden layer and output layer were computed and shown below in Equations (7.18) and (7.19) respectively. The output neuron yielded the predicted glucose value for the given PH. The proposed method was tested with PHs of 30 and 60 minutes. The RMSE
between the actual value and the predicted output were calculated for performance validation and verification.

\[
W_{i,j} = \begin{bmatrix}
0.488 & -0.341 & 0.707 & -0.914 & 0.896 & 0.564 \\
-0.027 & 0.083 & 0.756 & 0.186 & 0.511 & 0.229 \\
0.324 & 0.493 & -0.185 & 1.18 & 0.931 & -0.687 \\
1.183 & 0.324 & -0.101 & 0.486 & 0.196 & 0.542 \\
-0.303 & 0.823 & 0.229 & -0.471 & 0.763 & 0.681 \\
-0.832 & 1.321 & -0.677 & 0.268 & 0.398 & 0.409 
\end{bmatrix} \quad (7.18)
\]

\[
V_{i,j} = \begin{bmatrix}
0.981 \\
0.763 \\
0.062 \\
-0.734 \\
0.228 \\
1.107
\end{bmatrix} \quad (7.19)
\]

### 7.5 PERFORMANCE EVALUATION OF FNN

Though the interactions between the factors for glucose metabolism are complex, multidimensional, highly nonlinear, chaotic, stochastically and time variant series, the FNN model seemed to be a more suitable predictor. It could model the input-output behavior of glucose metabolism without knowing the involved explicit internal processes. Figure 7.4 shows the predictive capability of FNN with a PH of 30 minutes. A deviation was observed in the initial period of 0 to 70 minutes. However, in due course, the FNN was able to catch up the dynamics of BG accurately with a RMSE value of 6.4 mg/dL. Figure 7.5 depicts the prediction of FNN with a PH of 60 minutes. It was clearly demonstrated that the trend of BG variations were well followed by FNN with minor deviations, at instances of 300 minutes and 650 minutes, in the range around 4 and 6 mg/dL respectively. The performance measure of 14.6 mg/dL RMSE was obtained in this scenario. Although this value was higher than earlier scenarios with PH = 30 minutes, the efficiency of FNN could be well understood from Figure 7.4, which has shown the
tracking of FNN in the same trend as the original signal. The FNN has also been applied for three days data sets in which the predictions were made in both 30 minutes and 60 minutes PH. A sample of predictions in 30 and 60 minutes PH were shown in Figures 7.6 and 7.7 respectively.

Figure 7.4 Prediction of FNN in 30 minutes PH

Figure 7.5 Prediction of FNN in 60 minutes PH
Figure 7.6  Prediction of FNN in 30 minutes PH for Three days data set

Figure 7.7  Prediction of FNN in 60 minutes PH for Three days data set
The performance of the proposed method, FNN, was analyzed with the data sets mentioned earlier. The proposed method has been compared with the recent work on prediction with neural network trained by LM algorithm. For ease of representation, here we would refer to the neural network trained by LM algorithm as the LMNN. The details of prediction error in terms of the average RMSE in mg/dL, with the SD for both FNN and LMNN models in the PHs of 30 and 60 minutes are given in Table 7.3. In the 30 minutes PH, the prediction error of FNN was three times lower than that of LMNN in all the data sets. The mean RMSE value of FNN was 6.4 mg/dL, whereas it was around 19 mg/dL for the LMNN model. For PH values of 60 minutes, the error of the LMNN was almost twice that of the FNN. In the 60 minutes PH, RMSE of FNN was around 14.3 mg/dL, while that of the LMNN model was 28 mg/dL.

Tables 7.4 and 7.5 show the respective average delays in minutes and the SD of the rising and falling trends. For PH values of 30 minutes, the delay of the FNN was around 5 minutes, whereas it was 7 minutes for the LMNN. This delay increased with increasing PHs. For PH of 60 minutes, the delay of the LMNN was twice greater than the FNN. i.e., LMNN made predictions with an average time delay of 18.8 minutes, while FNN had an average delay of 9 minutes. In both the models, the delay in the upward trend was lesser than that of the downward trend. This could be explained by the fact that, for physiological reasons, the upward trends are usually more rapid than the downward trends, and the upward-to-downward changes are usually marked than the downward-to-upward changes. For this reason, both the models exhibit greater delay in crossing the threshold in the downward trend.
Table 7.3 Performance comparison of FNN in terms of RMSE

<table>
<thead>
<tr>
<th>Data Set</th>
<th>RMSE (Mean ± SD) in mg/dL for Prediction Horizons of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30 minutes</td>
</tr>
<tr>
<td></td>
<td>FNN</td>
</tr>
<tr>
<td>Data Set 1</td>
<td>4.0 ± 2.2</td>
</tr>
<tr>
<td>Data Set 2</td>
<td>7.8 ± 2.0</td>
</tr>
<tr>
<td>Data Set 3</td>
<td>7.5 ± 4.2</td>
</tr>
</tbody>
</table>

Table 7.4 Time lag of FNN in upward trend

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Time lag (Mean±SD) in minutes for Prediction Horizons of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30 minutes</td>
</tr>
<tr>
<td></td>
<td>FNN</td>
</tr>
<tr>
<td>Data Set 1</td>
<td>3.2 ± 2</td>
</tr>
<tr>
<td>Data Set 2</td>
<td>6.8 ± 3.3</td>
</tr>
<tr>
<td>Data Set 3</td>
<td>5.3 ± 1.5</td>
</tr>
</tbody>
</table>

Table 7.5 Time lag of FNN in downward trend

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Time lag (Mean±SD) in minutes for Prediction Horizons of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30 minutes</td>
</tr>
<tr>
<td></td>
<td>FNN</td>
</tr>
<tr>
<td>Data Set 1</td>
<td>8.0 ± 2.4</td>
</tr>
<tr>
<td>Data Set 2</td>
<td>9.4 ± 3.5</td>
</tr>
<tr>
<td>Data Set 3</td>
<td>10 ± 2.3</td>
</tr>
</tbody>
</table>

The prediction models could perform better in the upward trend than in the downward trends. The reason for higher time lag in the downward trend could be explained as follows. The glucose metabolism in human is
such that when the glucose level in blood falls, the glucose that has been stored in the liver would be released by the process of glycogenolysis, which leads to unexpected variations in the BG values. Hence, tracking of this unforeseen BG dynamics results in more time lag. Due to this reason, the prediction models exhibited greater overshooting in the upward-to-downward zones and the subsequent downward trend was dogged by this effect, thus, both the models exhibited a greater delay in crossing the threshold in the subsequent downward trend.

The efficiency of FNN could be evaluated with MSE since the back propagation process iteratively searches for a set of weights \( W \), which minimizes the error function \( E \) over all training patterns \( X^p \).

\[
E(W^*) = \min_w \{ \sum E^p(W, X^p) \}
\]  

(7.20)

Where \( E^p = \left( \frac{1}{2} \right) \sum (d^p_k - o^p_k)^2 \)  

(7.21)

MSE function is often chosen, because of its statistical properties and non-negative characteristic. It is a differentiable function that penalizes large errors more than small ones. When the training is complete, the network performance has to be checked. Cross validation is a standard tool in statistical prediction. Basically, the available data are sub divided into two sets, one for parameter estimation and the other for performance assessment. In ANN, the data sets were divided into three parts so as to have estimation, validation and testing. The network training parameters could be obtained from the training record. The value of the performance function i.e., the performance progress was plotted with ‘plotperf’ command. The output of ‘plotperf’ command is shown in the Figure 7.8. The iteration in which the validation performance reached a minimum was 11. The training continued for 6 more iterations, before training stopped.
Regression shows the relationship between the outputs of the network and the targets. If the training is perfect, the network outputs and targets would be exactly equal, but it is rare practically. If R=1, there would be an exact linear relationship between the outputs and targets. If R is close to zero, then there would no longer be a linear relationship between the outputs and targets. Here, the training data have indicated a good fit. The validation and test results have shown R values > 0.9. The efficiency of FNN in training process could be analyzed with the results given in Table 7.6.

Table 7.6 Efficiency of FNN

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Validation</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Set 1</td>
<td>Data Set 2</td>
<td>Data Set 3</td>
</tr>
<tr>
<td>MSE</td>
<td>0.167</td>
<td>0.184</td>
<td>0.211</td>
</tr>
<tr>
<td>Correlation Coefficient, R</td>
<td>0.92</td>
<td>0.95</td>
<td>0.95</td>
</tr>
</tbody>
</table>

The analyses of all the data sets were done through training, validation and testing phases. The MSE was found to be less than 0.25 in all the three scenarios and the correlation coefficient was nearer to or greater than
0.9, which confirmed a close relationship between the outputs and training. The performance of FNN was compared with GD NN and LMNN in terms of the factors given below in Table 7.7. The FNN algorithm had good ability to predict the future values with a high correlation coefficient, both in training and testing. The training errors were on an average of 0.168 in the training phase and 0.1465 in the testing phase.

The performance of the scenarios mentioned above were evaluated in terms of average number of iterations for the training phase and average MSE and percentage of correct predictions in the testing phase. The stopping criterion for the network training has been fixed as the MSE of 0.01. The results were compared and summarized in Table 7.7. Training of the FNN with extracted features has made it possible to meet the performance goal in lesser iterations, compared to GD and LM algorithm trained networks. Though LM algorithm has the advantage of having faster learning capability, its drawback would be the large memory requirement due to the computation of the Jacobian matrix. The size of the Jacobian matrix is $Q \times n$, where $Q$ is the number of training data sets and $n$ is the number of weights and biases in the network. As mentioned in section 7.4, the FNN has $6 \times 6$ weight vector between the input and hidden layer, and $6 \times 1$ weight vector between the hidden layer and output layer. For the applied large training data set and the specified weights, the LM algorithm needs a computational memory of 89Mb, whereas FNN needs a memory requirement of 56 Mb.
Table 7.7 Performance comparison of FNN with GD and LM algorithm

<table>
<thead>
<tr>
<th></th>
<th>Average number of Iterations in Training</th>
<th>Average MSE</th>
<th>% of Correct Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD NN</td>
<td>189</td>
<td>0.3877</td>
<td>72.0</td>
</tr>
<tr>
<td>LM NN</td>
<td>53</td>
<td>0.0549</td>
<td>90.5</td>
</tr>
<tr>
<td>FNN</td>
<td>55</td>
<td>0.0458</td>
<td>92.3</td>
</tr>
</tbody>
</table>

However, in the training phase, FNN has made the learning in a speed similar to LMNN and provided even better results. The average MSE was only 0.0458 mg/dL, which resulted in predictions of 92.3% accuracy. The GD method required longer training time than the other two. FNN had comparatively taken almost nearer number of iterations in training, as that of LMNN. This is an appreciable result with lesser computational memory requirement, thanks to the adaptation provided by the extracted features.

The learning rate parameter ‘$\alpha_k$’ determined the size of weight adjustments made at every iteration and had influenced the rate of convergence. The error surfaces that have broad flat shapes near minima require larger learning rate values, while steep narrow shapes near minima require small values, to avoid over shooting of the solution. If the chosen value is too large for the error surface, the search path will oscillate about the ideal path and converge more slowly or even could diverge. On the other hand, if the learning parameter value is too small, the progress would be in small steps, significantly increasing the total time to converge. These hurdles have been overcome by estimating the learning rate parameter value from the extracted features with the least square method of coefficients. The activation functions mentioned in Equations (7.4) and (7.5) which were customized to fire with learning rate, enabled FNN to make 92.3% accurate predictions. The
prediction performance could still be improved by training the network with large data sets and trials.

![Graph showing MSE error values](image)

**Figure 7.9 Training of FNN**

The graph in Figure 7.9 shows the fall in error values from 0.1926 to 0.0096, with an increase in number of iteration during the training process. This work has attested the effectiveness of training the feed forward neural network with the extracted features of inputs for predicting the future values in continuous glucose monitoring sensor data. The learning rate parameter derived from the extracted features, demonstrated the efficiency in computing the weights of neurons in back propagation. This could be well understood from the performance evaluation between FNN, GDNN and LMNN as listed in Table 7.7. From the trials made, it could be observed that ninety percent of hypoglycemic occurrences were detected by FNN even in 60 minutes PH, which was more appreciable, compared to previous works. From the results shown in Table 7.7 it could be observed that with a simple network structure, the proposed method has exhibited promising results i.e 2% higher than the LMNN method. The computational time for training was also acceptable with higher speed processors available to date. In future, the PH could be extended, so as to track the changing metabolism. Instead, 60 minutes PH would be
sufficient for a diabetic person to take necessary action with a hypoglycemia alert.

The sigmoid function used for activation in LMNN was a globalized function, whereas the localized feature set in FNN has increased the learning efficiency. The back propagation used in LMNN might get trapped in local minima. When the network was unable to converge to a global minimum, the prediction accuracy would deteriorate. This could be the reason for the less promising results of the earlier approach. Exponential functions were better approximated by radial basis function neural network. However, to predict the nonlinear dynamics of CGM time series, the specialized activation function with extracted feature set could be the best choice as proved by the results of this work.

7.6 SUMMARY

In this approach of prediction a multiobjective model has been presented, which included mean, variance, skewness, kurtosis and approximate entropy for the prediction of future values of CGM time series and compared the performance with traditional models. In the next chapter, the forecasting of glucose values have been tried by means of customized ANFIS with minimum number of input features.