Analysis of Feedforward and Recurrent Neural Network in Forecasting Foreign Exchange Rate

Sanju Singh Saini¹, Omkar Parkhe² & Prof. T.D.Khadtare³
1²³ Dept. of Information Technology, Sinhgad Institute of Technology and Science, Pune, India

Abstract: The forecasting of noisy, volatile and unstable natured foreign exchange rate has been challenging for all the researchers and business practitioners. The Artificial Neural Network has proved to be a very powerful tool for performing predictions and forecasts making it applicable for forex forecasting. In this paper we attempt to compare the performances of feed forward neural network and recurrent neural network used for forecasting the daily exchange rate of Indian Rupee against four base currencies namely, British Pound, US Dollar, Euro and Japanese Yen. This paper presents a detailed analysis over the network parameters, their role in training and their effect on the results. Overall both the networks gave good results for all the currencies with the recurrent network giving better results than feedforward.

Introduction

The foreign exchange market (Forex) is a global financial market where trading of currencies is done. The trading includes exchange of currency at current or pre-determined rate. The volume of trading done at Forex is largest with over average trade of $5.3 trillion a day making it world’s largest market. In India the volume of trade with foreign country has been steadily growing from past few years resulting in the growth of Forex. The amount of trade done every day in Forex makes it clear that any trader on Forex would like to have a much accurate forecast of the exchange rate for a trade done using a pre-determined rate. The literature shows that forecasting the exchange rates can be done with proper training of neural network.

The forecasts of the existing statistical and econometric models are not good enough and do not give accurate results. Even the traditional time series models such as Autoregressive Moving Average model (ARMA) and Autoregressive Integrated Moving Average model (ARIMA) used for non-linear time series forecasting like exchange rate, do not give good results as they make an assumption of an existence of a linear relationship among the input and the output. Thus, even today forecasting exchange rate is a big challenge for all the time series researchers and practitioners around the world.

The Artificial Neural Network (ANN) has proved to be a very powerful tool for performing predictions and forecasts. The literature shows that ANN has been successfully implemented in forecasting exchange rates. They have many advantages over the other models. First, they can approximate any nonlinear function without any prior information about the data. Second, ANNs are data-driven and self-adaptive method for forecasting and prediction.

In this paper we have forecasted the exchange rate of Indian Rupee (INR) against four base currencies namely, British Pound (GBP), US Dollar (USD), EURO (EUR) and Japanese Yen (JPY), using both feedforward and recurrent neural network. The motivation for choosing these currencies was the volume of trades done in these currencies i.e. more than 75% of all Forex trades. We have also analyzed and compared the results and performance of both the networks using different parameter values. Step-wise forecasting method was used to forecast the exchange rates with the prediction horizon of one day.

Literature Survey

Let us take a brief look at a few papers addressing the same problem of forecasting Forex using ANN.

In [1] the authors have forecasted the exchange rates of Japanese Yen, Swiss Frank, British Pound and EURO against US Dollar using recurrent neural network. The authors preprocessed the data collected in order to remove any possible correlations. The forecasts gave good results with approximately 80% accuracy and $R^2$ error in the range of 0.65 to 0.69. Their results suggest that neural network can be successfully implemented and used practically.
In [2] the authors performed Forex forecasting of US dollar, EURO, GBP and Japanese Yen against Nigerian Currency (Naira) using feed forward neural network. A multilayer perceptron was trained using backpropagation algorithm along with a Sigmoid Activation function. The authors compared their network’s result with Hidden Markov Model (HMM) of other author using mean squared error and standard deviation. On comparing the performances it was observed that the authors network outperformed the HMM with 81% accuracy.

In [3] the researchers have performed forecasts of weekly Indian rupee against US dollar using a hybrid model comprising of the ANN and ARIMA. The time series data has been pre-processed using the ARIMA model and its output has been fed to the ANN for forecasting. A comparison of results of their hybrid model, ANN, linear autoregressive model and random walk model was presented by the authors. Their study shows that the hybrid model did perform better than the linear autoregressive and random walk models and produced similar results to that of the ANN.

In [4] the authors have implemented both feed forward and recurrent neural networks for predicting the exchange rate of Euro and US Dollar against the Romanian Currency (RON). The data was been normalized and applied over a logarithmic function before feeding it to the networks in order to remove any correlations between the input and output. The comparison was done between a simple Elman Jordan architecture based recurrent neural network which was trained using the Extended Kalman Filter (EKF) method and a feed forward neural network which was trained using the Backpropagation algorithm. On comparison it was observed that the recurrent neural network clearly outperformed the feedforward neural network.

In [5] the authors have forecasted exchange rates of British pound, Canadian dollar, Deutsche mark, Japanese yen, and Swiss franc against US dollar using both feedforward and recurrent neural network selected on the basis of predictive stochastic complexity (PSC) criterion. The authors have trained the networks using recursive Newton algorithm and non-linear least square method for computing the PSC criterion and performing forecasts, respectively. ARMA model was also been built as a random walk model for comparing its results with the network results. Even though the neural networks gave them mixed results, but still they always produce good results than the random walk models.

The authors have built and compared the performances of neural network and a linear model for the forecasting the exchange rate of US dollar against the Deutsche Mark (DM) in [6]. They performed forecasting of the exchange rate on monthly as well as weekly data using both the methods. On comparing the results it was observed that both the methods gave similar results for monthly data but the neural network outperformed the linear model in case of weekly data.

In [7] the authors have forecasted the exchange rate of Turkish Lira (TL) against US dollar using Feedforward Neural Network. The authors also built traditional times series models, ARIMA and autoregressive conditional heteroscedasticity (ARCH), and compared their results with the neural network. The results show that ANN outperformed both the traditional time series models giving more accurate results, calculated in terms of RMSE, for both training as well as testing data.

**Methodology**

To perform the analysis between feed-forward and recurrent neural network in financial time series prediction, we chose foreign exchange rate forecast system. The system forecast Indian Rupee (INR) with four base currencies, namely American Dollar, Japanese Yen, British Pound and EURO. In the study we implemented feed-forward and recurrent neural network. And analyzed both the neural network on different performance criteria.

**Data Collection**

For the purpose of forecasting the exchange rates we collected seventeen years of data from 25/08/1998 to 08/10/2015 available on the Reserve Bank of India (RBI) website. The data is in the form of daily averages of the exchange rate for the period between two closing hours as shown in the below figures. The dataset was divided into two parts: training (80%) and testing (20%).

**Figure 1. USD/INR**

**Figure 2. GBP/INR**
Normalization of Data

Since the activation function chosen is sigmoid, which gives output in the interval \([0,1]\) it was necessary to normalize the data into this interval. The normalized data was fed as input to the neural network and results were calculated after denormalization of data.

Artificial Neural Network Architecture

After the data gathering, analysis and normalization, the architecture of FFNN and RNN were decided by performing analysis with different values of parameters. The initial weights were set to random values between \([-1,1]\) and Sigmoid Activation function was used to transform the input.

First, we designed a feedforward neural network, a multilayer perceptron with the three layers: input, output and hidden with 30, 15 and 1 neurons, respectively. The network was trained using the Backpropagation learning algorithm.

Second, we designed a simple Elman Jordan Recurrent Neural Network consisting of four layers: an input layer, two hidden layers and an output layer. The input layer consisted of 30 neurons, hidden layers consisted of 15 neurons each and the output layer consisted of a single neuron. The network was trained using the Backpropagation Through Time learning algorithm. The past 30 instances were fed as input to the networks and they would forecast the value of next instance.

Training the Neural Network

Training the Feedforward Neural Network using Backpropagation Learning Algorithm

Backpropagation Learning Algorithm is the most popular and widely used algorithm for training the feedforward neural network developed by Werbos in 1974. It is a powerful, and very widely and successfully used training algorithm. The goal of the algorithm is to obtain an optimal solution using gradient descent and attaining the minimal error.

Backpropagation algorithm is a supervised learning algorithm in which both the input and output are been provided to the network. The backpropagation learning algorithm consists of two phases: forward propagation phase and back propagation phase. Suppose we have \(n\) instances for training the network: input vector \(X = (x_{11}, x_{12}, \ldots, x_{in})\) and output vector \(Y = (y_{11}, y_{12}, \ldots, y_{in})\) for \(1 \leq i \leq n\).

In the first phase (forward-propagation phase), \(X_i\) is fed into the input layer, and an output vector \(\hat{Y} = (\hat{y}_{11}, \hat{y}_{12}, \ldots, \hat{y}_{in})\) is generated based on the current weight vector \(W\). The objective is to minimize an error function \(E\) defined as

\[
E = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{(y_{ij} - \hat{y}_{ij})^2}{2} \tag{1}
\]

through changing \(W\) so that all input vectors are correctly mapped to their corresponding output vectors.

In the second phase (back-propagation phase), a gradient descent function is used to the weight space, \(W\), in order to find the optimal solution. The direction and magnitude change \(\Delta w_{ij}\) can be computed as

\[
\Delta w_{ij} = -\frac{\delta E}{\delta w_{ij} \epsilon} \tag{2}
\]

where \(0 < \epsilon < 1\) is a learning parameter controlling the algorithm’s convergence rate.

The sum squared error calculated by Equation (1) is propagated backwards from the output layer to the input layer in the second phase. The weights between the neurons of two consecutive layers are adjusted as the error is propagated backwards. Both the phases are executed during each iteration of every epoch in the backpropagation algorithm until \(E\) converges or the epoch limit is reached.

After computing all partial derivatives the network weights are updated in the negative gradient direction. A learning constant defines the step length of the correction. The updates for the weights are given by

\[
\Delta w_{ij}^{(2)} = -\epsilon o_{i}^{(1)} \delta_{j}^{(2)} \quad \text{for} \quad i=1,..., k+1; \quad j=1,..., m \tag{3}
\]

And

\[
\Delta w_{ij}^{(1)} = -\epsilon o_{i} \delta_{j}^{(1)} \quad \text{for} \quad i=1,..., n+1; \quad j=1,..., k \tag{4}
\]

Where \(o_i\) is the output of layer \(i\) and \(\delta_j\) is the error of layer \(j\).
It is very important to update the weights as the error is been propagated backwards because only then the network will be able to learn and uncover the hidden patterns in the input data.

Training Recurrent Neural Network using the Backpropagation Through Time Algorithm (BPTT)

The BPTT learning algorithm used to train the recurrent neural networks is essentially an extension of the backpropagation learning algorithm. The training data given as input to the BPTT algorithm is in the form of \( <X, Y> \) where \( X \) is the input vector \( <x_0, x_1, \ldots, x_n> \) and \( Y \) is the output vector \( <y_0, y_1, \ldots, y_n> \).

**Figure 9. BPTT Unfolding Through Time**

Firstly, the BPTT algorithm unfolds the recurrent neural network through time as shown in the above figure. The recurrent neural network is unfolded or converted into two or more feedforward neural networks as per the depth of the recurrent network. In the above figure the recurrent neural network is been unfolded into three feedforward neural networks for the depth = 3.

Secondly, the training of the unfolded recurrent network is done in the same way as that of the backpropagation algorithm in the feedforward neural networks. On unfolding the recurrent network each feedforward network is been trained from left to right. The feedforward network at each unfolding level will be trained starting from level 1 to level 3 in above example. Each unfolded feedforward network, in the forward propagation phase, accepts the output of the previous network and the input vector \( X \), and processes it to produce its output. In the backward propagation phase the error is propagated backwards from the highest level unfolded feedforward network to the lowest level network. The final output of the network will be obtained from the last level feedforward network after the complete training of the network.

**Result evaluation and performance comparison**

After development of the networks they were tested and their performance was evaluated. In this evaluation, the trained neural network is evaluated against the unknown testing data. This evaluation was done to see whether the neural network generated by training, forecast accurate result or not. On evaluating the performances of both the networks, the results were compared.

**Results and Discussion**

Both the networks were tested using sum of squared error (SSE) and their results were compared and analyzed using various network parameters. A detailed analysis was carried out using different values of learning rate, momentum and epochs.

The number of epochs for which the network is been trained plays an important role in the forecasting. The performance of both networks were been analyzed by varying the epoch size. It is been observed that on training the network with more number of epochs the network learns the pattern in the data very well giving more accurate results.

The learning rate is a critical factor in the training of the neural networks. On training with a high (0.30) learning rate, it is observed that the network ends up going into a local minima. On the other hand, by keeping a low learning rate (0.10) the network learns slowly but tends to reduce the possibility of ending up in a local minima.

Momentum is another important factor in the training of the neural network which helps to smooth out of the variations or unusual conditions present in the training data. On keeping a high momentum (0.70) with high learning rate the network tends to pass the global minima, while on keeping a lower momentum for complete or half epoch size, the network tends to learn quite well and give better results.

The table present at the end of the paper shows the results of both the networks obtained on training them with different parameter values. The table consists of the network parameters namely, momentum, learning rate and epoch used for retrieving the results specified in the table. Overall both the networks gave good results with not much SSE for all the four exchange rates.

**Conclusion**

In this paper, we have compared the performance of the feedforward and recurrent neural network based on their results on unknown test data. On testing the networks, it has been observed that both the neural networks performed well on forecasting the exchange rates for all four currencies as specified in table 1. But on comparison of the networks, we found that the recurrent neural network gave better
results than the feedforward neural network. It is clear from the results that the ANN can successfully forecast the exchange rate with good accuracy and thus, can be used practically. And also there is a lot of scope for further research in forecasting of the time series using neural network because of its ability to approximate any continuous function.

Acknowledgements

We would like to thank our college principal Prof. S.N. Mali for his support and encouragement for doing research.

References


Table 1. Results of FFNN and RNN

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Epoch</th>
<th>Momentum</th>
<th>Learning Rate</th>
<th>Currency</th>
<th>FFNN</th>
<th>RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-5000</td>
<td>0.8</td>
<td>0.3</td>
<td>USD</td>
<td>151.65</td>
<td>143.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBP</td>
<td>318.97</td>
<td>375.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>363.49</td>
<td>323.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>433.05</td>
<td>382.61</td>
</tr>
<tr>
<td>2</td>
<td>1-10000</td>
<td>0.8</td>
<td>0.3</td>
<td>USD</td>
<td>130.48</td>
<td>137.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBP</td>
<td>365.78</td>
<td>354.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>292.25</td>
<td>302.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>407.47</td>
<td>372.31</td>
</tr>
<tr>
<td>3</td>
<td>1-10000</td>
<td>0.1</td>
<td>0.1</td>
<td>USD</td>
<td>132.03</td>
<td>130.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBP</td>
<td>383.48</td>
<td>351.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>297.17</td>
<td>263.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>371.57</td>
<td>339.67</td>
</tr>
<tr>
<td>4</td>
<td>1-10000</td>
<td>0.7</td>
<td>0.1</td>
<td>USD</td>
<td>128.84</td>
<td>128.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBP</td>
<td>363.18</td>
<td>346.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>291.05</td>
<td>269.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>380.21</td>
<td>329.18</td>
</tr>
<tr>
<td>5</td>
<td>1-5000</td>
<td>0.7</td>
<td>0.1</td>
<td>USD</td>
<td>132.86</td>
<td>124.01</td>
</tr>
<tr>
<td></td>
<td>5001-10000</td>
<td>0.1</td>
<td></td>
<td>GBP</td>
<td>381.49</td>
<td>353.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>289.59</td>
<td>267.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>399.89</td>
<td>330.05</td>
</tr>
<tr>
<td>6</td>
<td>1-5000</td>
<td>0.7</td>
<td>0.1</td>
<td>USD</td>
<td>128.72</td>
<td>131.29</td>
</tr>
<tr>
<td></td>
<td>5001-10000</td>
<td>0.1</td>
<td></td>
<td>GBP</td>
<td>365.86</td>
<td>357.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EUR</td>
<td>287.90</td>
<td>283.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JPY</td>
<td>384.26</td>
<td>344.51</td>
</tr>
</tbody>
</table>