USD Exchange Rate Forecasting using ARIMA, MLP and Recurrent Neural Networks

Bhargavi K
M.Tech. Student Dept. of Computer Science Engineering
BTL Institute of Technology & Management
Bangalore, India

***Abstract*- The motivation of this paper is to investigate the use of Multi-Layer Perceptron (MLP) and Recurrent neural network(RNN) model and Auto-Regressive integrated Moving Average Model (ARIMA) model architectures to forecast the dollar exchange rate. MLP and RNN will be trained by backpropagation algorithm. The prediction accuracy of all three methods are compared based on mean absolute error .**

Index Terms - Neural Network (ANN); Auto-Regressive integrated Moving Average Mode; Multi-Layer Perceptron; mean absolute error; Recurrent Neural Network.

I. INTRODUCTION

The term of Neural Network (NN) originates from the biological neuron connections of human brain. The neural network will be trained by previously generated data and the model will be generated then the model will be use to forecast the future data. The artificial NNs are computation models that embody data-adaptive learning and clustering abilities, deriving from parallel processing procedures [1]. The NNs are considered a relatively new technology in Finance market, Because of high potential will be used in various applications. However, their practical restrictions and differing empirical evidence lead to scepticism on whether they can outperform existing traditional models.

The motivation of this paper is to investigate the use of Multi-Layer Perceptron (MLP) and Recurrent neural network(RNN) model and Auto-Regressive integrated Moving Average Model (ARIMA) model architectures to forecast the dollar exchange rate. MLP and RNN will be trained by backpropagation algorithm. The prediction accuracy of all three methods are compared based on mean absolute error.

Prediction of various currency exchange rates is influenced by many factors such as economical, political and psychological and hence it is a complex task to forecast these values by classical methods. The data is predicted by the technical scientists by tracing patterns that are mostly achieved from the study of charts that describe past data. As the percentage of in the training samples increases correspondingly the prediction accuracy is decreases. To overcome this problem various neural network techniques are used.

II. LITERATURE REVIEW

The most common NN architecture is the MLP and seems to perform well at time-series financial forecasting [2], although the empirical evidence can be contradictory in many cases. For example, Tsaih et al. [3] attempt to forecast the S&P 500 stock index futures and in their application Reasoning Neural Networks perform better than MLPs.

A classical methodology for carrying out this prediction is presented by Bowerman [4], which uses time series (TS) of the data. Tang [5] compared several stream flow forecasting models. Tang found that among TS methods with long memory, Box & Jenkins methods had the best performance. However, it was also reported in [5] that among TS methods with short-term memory, ANNs had the best performance. Kisi in 2005 [6] reported a better performance of ANNs to forecast streamflows than ARIMA models. Wen [7] also obtained a better performance with ANNs than ARIMA methods. In this paper a new real case of streamflow forecasting is presented which ARIMA Vs ANN models are compared error rate of Forecasting on stream of daily closing rates of USD.

III. THE USD EXCHANGE RATE AND PREPROCESSING

The European Central Bank (ECB) publishes a daily fixing for selected USD exchange rates: these reference closing rates are based on a daily consideration procedure between central banks within and outside the European System of Central Banks. The reference exchange rates are published both by electronic market information providers and on the ECB's website shortly after the attentiveness procedure has been completed. Based on the reference rate, many financial institutions are ready to trade at the EUR fixing.

In this paper, we predict the EUR/USD of some period using the previous data. we further divided our in-sample dataset in two subparts. One is train data and test data. Once the neural network is trained the model will be generated. then generated model is tested by using test data to check weather the model is fitted properly or not. The graphical plot of the input data is shown below in figur.1.

On other side the input data will be fitted to ARIMA model and then prediction will be calculated.

The prediction error is calculated in order to assess the adequacy of each model in terms of how well each one is forecasting. This is done by calculating Mean Absolute Error(MAE) for both predictions.

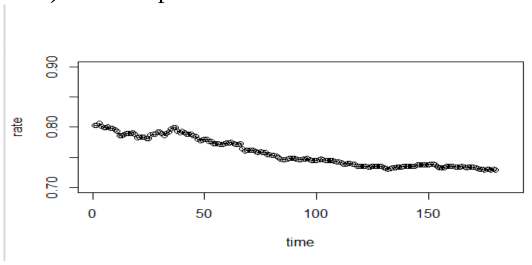


Figure :1 plot of input data set

The equation of MAE is

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m} |y_{t+i} - \hat{y}_{t+i}| \qquad (1)$$

Where

- y_{t+i} : Observed data in the series belonging to the prediction set.
- $\hat{y}_{t+i}$: Values predicted by model

IV. AUTO-REGRESSIVE INTEGRATED MOVING AVERAGE MODEL (ARIMA)

The ARMA model is based on the assumption that the current value of a time-series is a linear combination of its previous values plus a combination of current and previous values of the residuals [8]. The input data suppose to be Stationary TS. When a TS is nonstationary, this issue can be solved by defining a new variable as follows:

$$Z_t = y_t - y_{t-1} \qquad (2)$$

Where t=2,3,..n

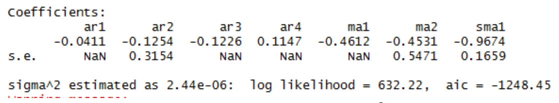
The ARIMA model embodies autoregressive and moving average components and can be specified as below:

$$Y_t = \varphi_0 + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p} + \epsilon_t - W_1\epsilon_{t-1} - W_2\epsilon_{t-2} - \ldots - W_q\epsilon_{t-q} \qquad (3)$$

Where:

- Y_t is the dependent variable at time t
- $Y_{t-1}, Y_{t-2}, \ldots Y_{t-p}$ are the lagged dependent variables
- $\varphi_0, \varphi_1, \ldots, \varphi_p$ are the regression coefficients
- ϵ_t is the residual term
- $\epsilon_{t-1}, \epsilon_{t-2}, \ldots, \epsilon_{t-q}$ are the previous values of the residual terms
- $W_1, W_2, \ldots, W_q$ are the residual weights.

The stationary data will be fitted to ARIMA model by using method arima() based on in-sample correlogram a restricted model is chosen as ARIMA(4,1,2). The figure 2 represents the obtained parameters with standard error from arima().



```
Coefficients:
          ar1      ar2      ar3     ar4      ma1      ma2     sma1
      -0.0411  -0.1254  -0.1226  0.1147  -0.4612  -0.4531  -0.9674
s.e.      NaN   0.3154      NaN     NaN      NaN   0.5471   0.1659

sigma^2 estimated as 2.44e-06:  log likelihood = 632.22,  aic = -1248.45
```

Figure 2: output parameters of ARIMA

Substituting computed parameters in equitation (3) can be written as

$$Y_t = 0.0411 - 0.1257y_{t-1} - 0.1223y_{t-2} + 0.1147y_{t-3} - 0.4612\varepsilon_{t-1} - 0.4531\varepsilon_{t-2} \qquad (4)$$

Finally in the last step is to test the resulting ARIMA model by the inspection of the one-step prediction residuals {εt}. This was done by a statistical classical test based on the autocorrelation function of the residuals.

Figure 3 shows forecasting results with ARIMA model while its prediction errors obtained from above model is shown in figure 4

V. NEURAL NETWORKS

Neural Networks exist in several forms in the literature. The most popular architecture is the Multi-Layer Perceptron (MLP) and Recurrent Neural network. A standard Neural Network has at least three layers. The first layer is called the input layer (the number of its nodes corresponds to the number of input variables). The last layer is called the output layer (the number of its nodes corresponds to the number of response variables).An intermediary layer of nodes is called hidden layer, separates the input from the output layer. Its number of nodes defines the amount of complexity the model is capable of fitting. In addition, the input and hidden layers contain an extra node called the bias node. This node has a fixed value of one and has the same function as the intercept in traditional regression models.

Normally, each node of one layer has connections to all the other nodes of the next layer. The network processes information as follows: the input nodes contain the value of the input variables. Since each node connection represents a weight factor, the information reaches a single hidden layer node as the weighted sum of its inputs each node of the hidden layer passes the information through the nonlinear activation function and pass it to the output layer if calculated value is above specified threshold.

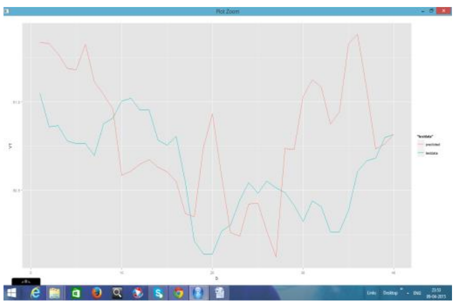
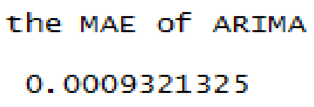


Figure 3: plot of observed data and predicted data of ARIMA

The process of training the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and trained by applying a learning algorithm called backpropagation Algorithm[9]. The learning algorithms imply tries to find those weights which minimize an Error Function. Since networks with sufficient hidden nodes are able to learn the training data as well as their outliers and their noise. it is critical to stop the training procedure at the right time to prevent overfitting this process is called ‘early stopping’.



```
the MAE of ARIMA

 0.0009321325
```

Figure 4:MAE of ARIMA

The network parameters are then estimated by fitting the training data using the above mentioned iterative procedure (backpropagation of errors). The iteration length is optimized by maximizing the forecasting accuracy for the test dataset.

A. *The Multi-Layer Perceptron Model(MLPs)*

MLPs are feed-forward layered NN, trained with a back-propagation algorithm. According to Kaastra and Boyd [34], they are the most commonly used types of artificial networks in financial time-series forecasting. The training of the MLP network is processed on a three layered architecture, as described above. A typical MLP model is shown in Figure. 5.

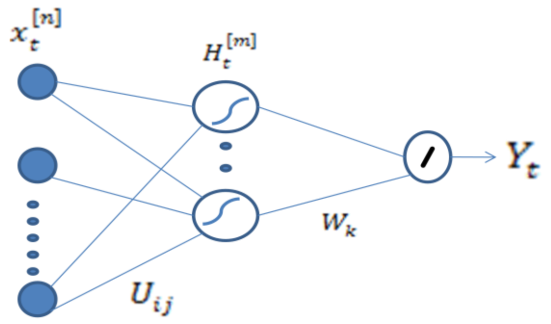


Figure 5: A single output, fully connected MLP model (bias nodes are not shown for simplicity).

Where:

- $x_t^{[n]}$(n=1,2,….,k+1)Z are the inputs (including the input bias node) at time t
- $H_t^{[m]}$(m=1,2,…, j+1)Z are the hidden nodes outputs (including the hidden bias node) at time t
- Y_t is the MLP output
- U_{ij} W_k are the network weights
- is a linear function
- F(x)=$\sum_i x_i$
- is the transfer sigmoid function
- S(x)= $\frac{1}{1+e^{-x}}$

The error function to be minimized is

$$E(c,w_j)=\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t(w_k,c))^2 \qquad (5)$$

with y_t being the target value. The evaluation of the MLP model selected comes in terms of trading performance. The time series data will be divided as Training data set: Consisting of 80% of the TS data used for training the neural network. Testing data set: Consisting of 20% of the TS data. These are the remaining data, once the training dataset were selected. This data set are used to evaluate the performance of the network. In this experiment the parameters are set as shown below to train the network

- Maximum number of iterations: 2000
- Error convergence allowed: 0.01
- Learning rate: 0.01
- adaptive learning rate: 1.05

The learning rate must be kept as small as possible, but in order to reduce the training time the adaptive learning rate should tuned bigger that one. The plot of actual data and predicted data of the neural network is shown below in figure 6 and error in figure7.

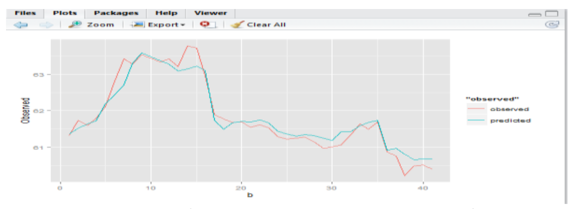
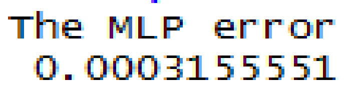


Figure 6: The plot of actual data and predicted data of the MLP



```
The MLP error
 0.0003155551
```

Figure 7: The mean absolute error of MLP

B. *Recurrent Neural Network (RNN)*

The fundamental feature of a Recurrent Neural Network (RNN) is that the network contains at least one feed-back connection, so the activations can flow round in a loop. That enables the networks to do temporal processing and learn sequences, e.g., perform sequence recognition/reproduction or temporal association/prediction. Recurrent neural network architectures can have many different forms. One common type consists of a standard Multi-Layer Perceptron (MLP) plus added loops. These can exploit the powerful non-linear mapping capabilities of the MLP, and also have some form of memory. For simple architectures and deterministic activation functions, learning can be achieved using similar gradient descent procedures to those leading to the back-propagation algorithm for feed-forward networks. The architecture of recurrent neural network is shown below in Figure 8.

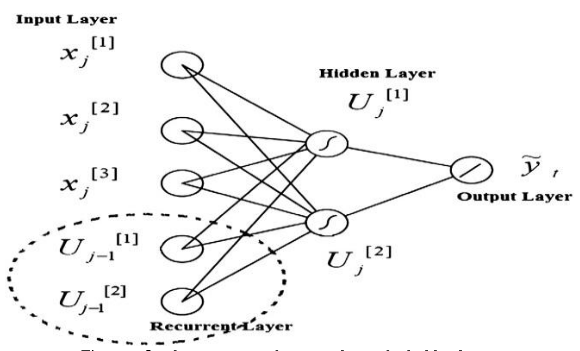


Figure . 8. Elman RNN with two nodes in the hidden layer.

Where:

- $x_t^{[n]}$ (n=1,2,…….,k+1), $u_t^{[1]}$, , $u_t^{[2]}$ Z are the RNN inputs at time t
- $d_t^{m[f]}$(f=1,2) and $w_t^{[n]}$(n=1,2,……,k+1) are weights of network
- $\tilde{y}_t$ is the RNN output
- $u_t^{[f]}$,f=(1,2) is the output of the hidden nodes at time t.
- is the transfer sigmoid function

  $$s(x)=\frac{1}{1-e^{-x}}$$
- is the linear function

  $$F(x)=\sum_i x_i$$

The error function to be minimized is

- $E(c, w_j)=\frac{1}{T}\sum_{t=1}^{T}(y_t-\hat{y}_t(d_{t,}w_t))\ 2$

The RNN architecture can provide more accurate outputs because the inputs are (potentially) taken from all previous values.

The evaluation of the RNN model selected comes in terms of trading performance. The time series data will be divided as Training data set: Consisting of 80% of the TS data used for training the neural network. Testing data set: Consisting of 20% of the TS data. These are the remaining data, once the training dataset were selected. This data set are used to evaluate the performance of the network. In this experiment the parameters are set as shown below to train the network

- Maximum number of iterations: 2000
- Error convergence allowed: 0.01
- Learning rate: 0.01
- adaptive learning rate: 1.05
- number of hidden nodes:19

The graphical representation of the plot of observed data and predicted data is shown in figure 9. And related MAE is shown in figure 10.

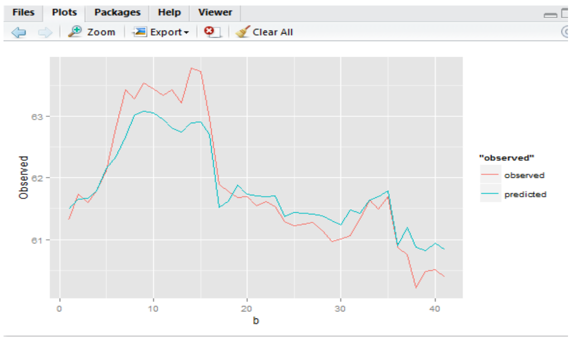
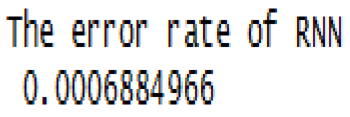


Figure 9: plot of observed and predicted data of RNN



```
The error rate of RNN
0.0006884966
```

Figure 10: Mean Absolute Error of RNN

VI. CONCLUSIONS

Three methodologies for dollar rate forecasting were analyzed in this paper: ARIMA and Multi-layer perceptron model and Recurrent Neural Network. In the case of MLP and RNN Back propagation algorithm is used to train the network. data were collected from a real case and they were used for training these models. ANN parameters were carefully selected experimentally. Experimental results showed that ARIMA obtained much better prediction results than ANN multi-layer perceptron model and Recurrent Neural Networks.

REFERENCES

[1] Krose, P.V.D. Smagt, An Introduction to Neural Networks, eighth ed, University of Amsterdam, 1996.

[2] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: results of a forecasting competition, Journal of Forecasting 1 (2) (1982) 111–153.

[3] R. Tsaih, Y. Hsu, C.C. Lai, Forecasting S&P 500 stock index futures with a hybrid AI system, Decision Support Systems 23 (2) (1998) 161–174.

[4] Bowerman B.L, O´Connell R.T. *Forecasting and Time Series: an applied approach*. Wadsworth, Inc. Third Ed. 1993.

[5] Zaiyong Tang, de Almeida, Chrys, Fishwick, Paul *A. Time series forecasting using neural networks*

[6] Ozgur Kisi, Daily River Flow Forecasting Using Artificial Neural Networks and Auto-Regressive Models, *Turkish J. Eng. Env. Sci.* 29, 2005, pp. 9-20.

[7] Wen Wang, *Stochasticity, nonlinearity and forecasting of streamflow process*, IOS Press,
August 2006

[8] C. Brooks, Introductory Econometrics for Finance, second revised ed, Cambridge
University Press, Cambridge, 2008.