International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 7, Issue 2, March - April 2018 ISSN 2278-6856

Time series forecasting using ARIMA and Recurrent Neural Net with LSTM network

AvinashNath[1], Abhay Katiyar [2],SrajanSahu[3], Prof. Sanjeev Kumar [4]

[1][2][3] Department of Information Technology, ABES IT from Dr. A.P.J. Abdul Kalam Technical University, NH24 Road, Ghaziabad, Uttar Pradesh, India

[4] Department of Computer Science, ABES IT from Dr. A.P.J. Abdul Kalam Technical University, NH24 Road, Ghaziabad, Uttar Pradesh, India

Abstract: ***Autoregressive integrated moving average (ARIMA) or Box-Jenkins Method is a popular linear model for time series forecasting over the decade. Recent research has shown that the use of artificial neural net improves the accuracy of forecasting to large extent. We are proposing the solution in order to extract both Linear and Non-Linear components in data. In this paper, we propose a solution to predict highly accurate results using an aggregation of ARIMA and ANN (Recurrent neural net) to extract Linear and Non-Linear Component of data respectively.***

Keywords: ARIMA, Box–Jenkins methodology, artificial neural networks, Time series forecasting, recurrent neural net, combined forecasting, long short-term memory.

1. Introduction

Time series forecasting is a major and one of the most important type of forecasting in which past observations of the same variable are collected and analyzed to develop a model describing the underlying relationship between the future and the past observation. Time series *forecasting* [1]comprises methods for analyzing time series data in order to extract meaningful statistics and features from the data, which may have correlating. These models are used to predict future values based on previously observed values.

1.1 ARIMA

Over the past few years ARIMA[2] models are, in theory, the most general class of models for forecasting a time series data, which can be made to be "stationary" by differencing. ARIMA model consists of two major parts in general, an autoregressive (AR) part and a moving average (MA) part and (I) stands for Integrated. Here the AR part involves regressing the variable on its own lagged values. The MA part involves modeling the error term as a linear combination of error terms occurring at various times in the past.

1.2 RNNs

RNN stands for Recurrent Neural Net. In our work, we are using bidirectional multilayer feedforward deep neural net with LSTM cell. The basic concept of RNNs is to extract sequential information. In a regular neural network, we assume that all inputs and outputs are not correlated to each other but for the sequential or time-series data that's a very bad approach, to begin with. If we want to predict the next word in a sentence we better know which words came before it. RNNs is a type of ANN(Artificial Neural Net) which is basically called recurrent because they perform the same task for every element of a sequence, having the output being depended on the previous computations in the process. Another way of understanding about RNNs is that they have a "memory" which captures information or relevant features that have been calculate so far in the computation process.

2. Proposed Methodology

In our solution, we are using ARIMA for forecasting the values and using artificial neural net for improving those forecasted value. The artificial neural net we are using is called a recurrent neural net, which has a long term short memory cell in it so the vanishing and exploding gradient problem can be handle.

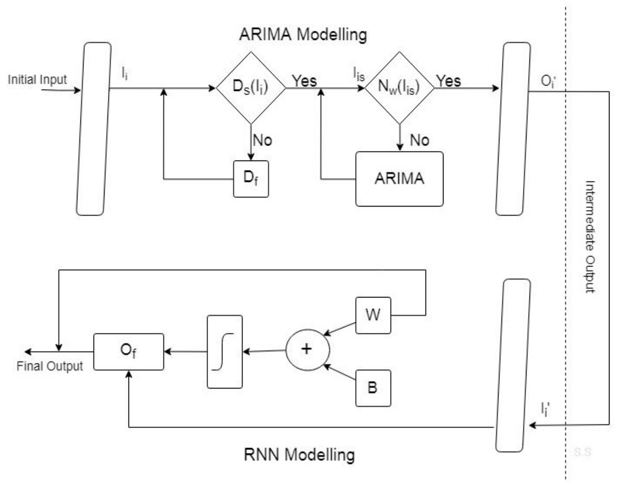

Figure 1: Combine Model of ARIMA and RNN

Variables in Fig: 1

I_i :	Initial Input	**B:**	Bias
D_f:	Differencing	**W:**	Weight
N_w:	White Noise	**+:**	Summation
O_f:	Final Output	**O_i' :**	Intermediate Output
ʃ:	Activation Function	**I_i' :**	Intermediate Input

In the figure 1 the full methodology has been shown where the initial inputI_i is insertedinto the ARIMA which is then converted into the intermediate output$O_i$and then that intermediate output is feed into the pre-trained model of RNN which improve the forecasted value and generate the final output O_f.

2.1 Model Description

Here we are using the cryptocurrency data-set for the time-series forecasting.
First, we areconverting the data set into time-series object. Then we need to pad it with the particular time interval (e.g. days, week, months etc.)based on our requirement.
This is one of the most critical steps in machine learning, whichfillsthe missing values, smooths noisy data, identifies or remove outliers, and resolve inconsistencies so that there should not be any unnecessary anomaly in the data.
Our entire model is divided into two different phases on the bases of component extraction. In first phase the linear component will be extracted using ARIMA modeling and in the second phase RNNs will be used in order to extract the Non-Linear component from the data. First phase focuses on forecasting the values and the second phase on improving those forecasted results by the neural net.

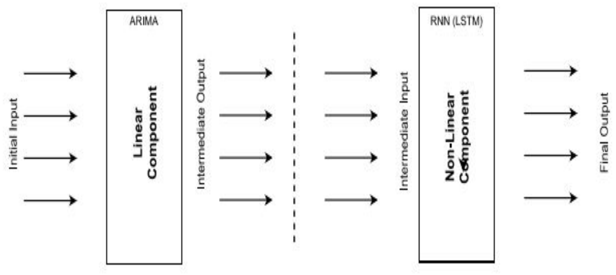

Figure 2: Two Phase Model Architecture

2.1.1 ARIMA Modeling

This is the first phase of our model where we are using ARIMA for modeling**a stationary time series. Stationary time-series is one whose properties do not depend on the time at which the series is observed. In order to achieve a stationary data, we need to do differencing.**

Differencing -help in stabilizing the mean and variance of a time series by removing changes in the level of a time series data, and it eliminates trends,seasonality and cyclical patterns. We are also using Dickey Fuller Test for checking whether the data is stationarity or not. There are various types of differencing methods. In our solution, we are using first order differencing.

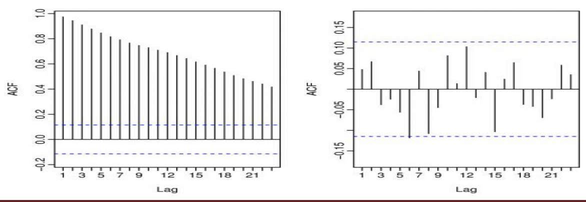

Figure 3: ACF plot of non-stationary as well as stationary data

In the Figure 3 there are two ACF plot .The left plot is the initial data with trend and seasonality and the right plot is without any of the seasonal and trend components.

In our solution we are using first order differencing for making our data stationary. The mathematical formula for computing first order differencing is given below:

$$y'_t = y_t - y_{t-1}$$

Here the $Y_{(t-1)}$is the previous observation so the final list will have $N-1$ values where N is the number of the initial entry.

Autoregressive Model – AR (*p*) refers to the autoregressive model of order *p*. Mathematically the AR (*p*) model can be written as:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

Where$\varphi_1, \dots, \varphi_p$ are model's parameters, $c$is the equation constant, and theε_t is the white noise.

Moving-average model –MA (q) refers to the moving average model of order q.Mathematically it can be written as:

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

Where $\theta_1, \dots, \theta_q$ are the model's parameters,μ is the expectation and the$\varepsilon_t, \varepsilon_{t-1}, \dots$are the white noise.

After finding the optimal ARIMA(p,d,q) value we have forecasted the output and stored into a file so that it can further be inserted into the Neural net as an input in second phase.

2.1.2 RNNs Modeling

In our work, we are using bidirectional multilayer feedforward recurrent neural net with LSTM network.
To understand RNNs we are explaining simple feedforward neural net. The basic formula for the feedforward deep neural net can be shown as.

$$activation\, f\big((weight\, w * input\, x) + bias\, b\big) = output\, y$$

Where f is the activation function that have the mathematical summation of biases b and the product of weighted matrix w and input vector. Thisbasic type of feedforward neural net where the data flows through the function being evaluated from x, through the intermediate

computations used to define f, and finally to generate the output y. These types of network have no recurrent feedback, which can improve the weighted matrix.
When these simple types of simple feedforward neural networks are extended to include feedback connections as an input to the network in order to improve, the weights are called recurrent neural networks (RNN). RNNs use the same computation (determinedby the weights, biases, and activation functions) for every element in the input series for several number of times. In mathematical terms, these equations define how an RNN evolves over time by the feedback it gains over time from the output of these neurons.

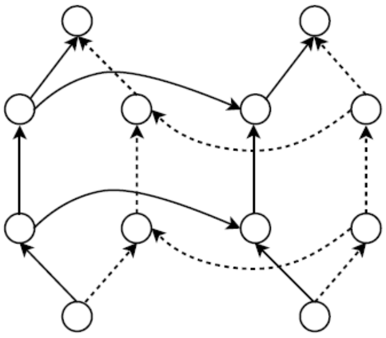
Figure4: Recurrent Neural Net

In our work, we are using gradient descent as an optimization algorithm so that we can iteratively moving in the direction of steepest descent from the entire domain. We use gradient descent to update the parameters of our model. In this, neural net parameters are referred as weights in the network.
Mathematically gradient can be calculated as:

$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b\,)) \end{bmatrix}$$

Where the cost function can be written as:

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

During the gradientback-propagation phase, the gradient signal can end up being multiplied by several number of times by the weight of the recurrent hidden layer which is unnecessary, which means that the weights in the transition matrix can have a strong impact on the learning process. In these neural net(RNN),if the weights in this matrix are small (or, more formally, if the leading Eigen value of the weight matrix is smaller than 1.0), it can lead to a situation called `vanishing gradients` problem where the gradient signal gets so small that learning either becomes very slow or stops working.It willmake the task of learning long-term dependencies in the data more difficult, which is our main objective. On other hand if the weights in this matrix are large (more formally, if the leading Eigen value of the weight matrix is larger than 1.0), in that case it can lead up to a situation where the gradient signal is so large that it can cause learning to diverge and cause problem. This is also called as `exploding gradients. To overcome this issue we are using a Long Short Term Memory` (LSTM) networks in RNN model. We have used LSTM in our solution to avoid the long-term dependency problem, which is the basic nature of LSTM and it helps in removing the vanishing and exploding gradient problem that exist in recurrent neural net. Remembering information for long periods of time is the very nature of LSTM network.

Mathematicallyit can be proved that, LSTM networks can remove the exploding and vanishing gradient problem. Let us assume that the hidden state in the neural net is ht at time step$t$. If we remove biases and inputs from the existing equation shown below.

$$ht = \sigma(wh_{t-1} * x + b_t)$$

We have,

$$ht = \sigma(wh_{t-1})$$

Then we can show that:

$$\frac{\partial h_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} w\sigma'(wh_{t'-k})$$

$$\frac{\partial h_{t'}}{\partial h_t} = w^{t'-t}\prod_{k=1}^{t'-t} \sigma'(wh_{t'-k})$$

In the above equation the $w^{t'-t}$is critical one. If in the above weight w is not equal to 1, it will either decay to zero exponentially fast in $t' -$ t order, or grow exponentially fast. This is our major problem.

In LSTMs, there is a cell where we have the cell states_t. The derivative of the cell state is in the form of:

$$\frac{\partial s_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} \sigma(v_{t+k})$$

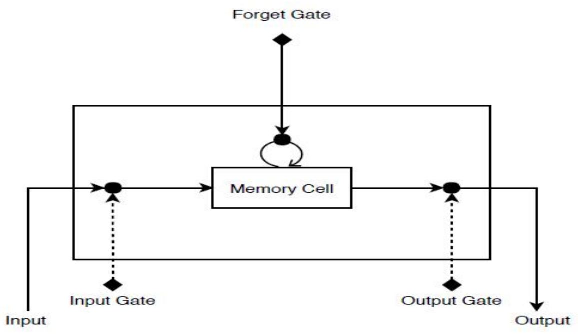

Figure 5: Long Short Term Memory

Here in the above equation thev_t is the input to the forget gate as shown in above block diagram. There is no exponentially fast decaying factor involved hence in our solution LSTM solves the problem of vanishing gradient and exploding gradient problem.

2.1.3 ModelOptimization

In this, we will try to change the value of $p, d\ and\ q$ in order to get the best forecasting result and then feed that output to the neural net for making better prediction.

We can also change the Activation function to the $sigmoidal$ or $tanh$ and biases with variable learning rate in order to get the best forecasting from the combined model of ARIMA and Recurrent neural net. Optimization can also be achieved by increasing the number of Hidden Layers and the number of epochs while training the Neural net.

RESULT AND CONCLUSION

Neural network is one of the vastest subject in the field of research and development. Many data scientists solely focus only on time series forecasting using neural network techniques. In our work, we have forecasted the output by both linear and non-linear model.

Table 1: Output of both model

Output Type	ARIMA(3,1,3)	ARIMA+ RNN
Forecasted Output	8215.62	8221.88
Actual Output	8263.84	8263.84

In the above Table 1 as it can beseen, that extracting both linear and non-linear component can leads to high accuracy.

Table 2: Measurement of CombineModel Accuracy

Model Validation	Output
MSE	16.236
RMSE	12.577
MAE	11.980

By the Table2,it can be seen that the validation score is not that bad and it can be improved to some extent. Therefore, it can be concluded that by using the combined forecasting approach with hyper tuning have much potential.
Neural Networks have much more advanced techniques in the field of forecasting. There is a lot of exciting research going on, around neural networks and if we talk about practical implementation then we can say that it can turn out to be the useful model for the forecasting.

REFERENCE

[1]Abraham, B. and Ledolter, J. (1983). Statistical Method for Forecasting, Wiley, New York, NY.
[2] Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). Time Series Analysis, Forecastingand Control, 3rd ed. Prentice Hall, Englewood Clifs, NJ.
[3] Box, G. E. P. and McGregor, J. F. (1974). &The Analysis of Closed-Loop Dynamic StochasticSystems;, Technimetrics, Vol. 16-3.
[4] Brockwell, Peter J. and Davis, Richard A. (1987). Time Series: Theory and Methods,
[5] Puyang Xu and DamianosKarakos and Sanjeev Khudanpur. Self Supervised Discriminative Training of Statistical Language Models.ASRU 2009.
[6] Denis Filimonov and Mary Harper. 2009. A joint language modelwith fine-grain syntactic tags. In EMNLP.
[7] Bengio, Y. and Senecal, J.-S. Adaptive Importance Sampling toAccelerate Training of a Neural Probabilistic Language Model.IEEE Transactions on Neural Networks.
[8] Morin, F. and Bengio, Y. Hierarchical Probabilistic Neural NetworkLanguage Model. AISTATS'2005.
[9] Tom´aˇsMikolov, Jiˇr´ıKopeck´y, Luk´aˇsBurget, Ondˇrej Glembekand Jan Cˇ ernocky´: Neural network based language models forhighly inflective languages, In: Proc. ICASSP 2009.
[10] T. Hain. et al., "The 2005 AMI system for the transcription ofspeech in meetings," in Proc. Rich Transcription 2005 SpringMeeting Recognition Evaluation Workshop, UK, 2005.
[11] Mahoney, M. 2000. Fast Text Compression with Neural Networks.

AUTHOR

Avinash Nath is an undergraduate Information Technology student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of Interest is Data science and Deep Learning.

Abhay Katiyar is an undergraduate Information Technology student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of interest is Data Science and Deep Learning.

Srajan Sahu is an undergraduate Information Technology student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of interest is Data Science.

Sanjeev Kumar received the B.Tech(CSE) and M.Tech(IT). He is currently pursuing PhD. from IIT ISM Dhanbad along with working asAssistant Professor in ABES IT. His area of specialization is Machine Learning.