International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-6, April 2020

Stock Prices Prediction with Recurrent Neural Networks

Middi Appala Raju, Venkata Sai Rishita Middi

Abstract: ***Data and Information is the base for making investment choices. Stock market is typically a place where shares of certain companies trying to raise their capital, are traded. With the availability of large amount of data and refinement methods, investors nowadays, are able to make rational investment decisions. Advancement in computational intelligence, use of AI in the form of Neural Networks has created a new basis for predicting stock prices. In this work, we have employed Recurrent Neural Networks to implement time series prediction. The Long Term Short Memory Architecture has been used as the network architecture to perform prediction on Apple Stock Prices. The implementation is done on Keras platform.***

Keywords ***: Activation, Neural Networks prices,prediction, stocks regressor, testing, performance***

I. INTRODUCTION

Stock Markets have become vibrant and dynamic with the integration of economies across the globe and participation of different investing classes, retails, financial Investors etc. in a variety of financial instruments. Market dynamics constantly change and rapidly alter the values of stock prices of different companies. Markets have become more and more efficient with the availability of data across different economies. Stock market analysts us a variety of techniques to predict stock prices. Technical Analysis is widely used over the years. With recent advancements in technology, AI and its techniques have taken precedence over earlier forms.

The ultimate aim of this project is to build a time series prediction system using Recurrent Neural Networks regressor. The results of the project are compared with the figure where stock price stock price of Apple was forecasted roughly 7 days in advance. The stock prices over a duration of 140 days is loaded as the training set. The paper is divided into five sections: (1) Loading the dataset and normalizing the range. (2) Breaking the time series into sequences (3) Segregating the test sets. (4) Building and Running the RNN Regression model. (5) Checking the model performance.

Revised Manuscript Received on April 30, 2020.
***** Correspondence Author
Middi Appala Raju*, Department of Management Studies, Christ(Deemed to be University), Bangalore, India. Email: middi.raju@christuniversity.in
Venkata Sai Rishita Middi, Department of Electrical and computer ENgineering, Brown University, Providence, India. Email: Venkata_sai_rishita_middi@brown.edu

II. LITERATURE REVIEW

A. Recurrent Neural Networks

Recurrent Neural Networks are a powerful and robust type of networks and belong to the most promising algorithms at the moment as they are the only ones with an internal memory. Because of the internal memory, RNNs are able to remember important things about the input they received which enables them to predict what is coming next. RNN are preferred algorithms for sequential data like time series, financial data which form a deeper understanding of a sequence and its context compared to other algorithms. In RNNs information cycles through a loop. In the decision making it takes into consideration the current input and also the learnings from inputs previously received.

B. Related Work

The review of Stock prices prediction has been discussed in [1] along with various techniques for stock market prediction. Multi-layer perception model, back propagation neural networks has been discussed for the forecasting of the stock market in [2]. Time series prediction has been built using Recurrent neural Networks in [4]. The Long Short term Memory architecture has been discussed in [5]. In [6], the investigation of the effectiveness of LSTM architecture has been performed. A hybrid model which is robust and novel has been proposed in [9]. In [8], the historical data of China stock market was converted into long sequences with 10 learning features.

III. PART 1: PRE-PROCESSING

A. Loading Data

It is very important to load the dataset. As mentioned earlier, we load the history of Apple stock prices over a duration of 140 days. We have used the matplotlib inline in order to display the output of plotting commands inline within Jupyter notebook.

B. Normalization Of Input Data

It is always a good practice to normalize the range of time series. Generally, normalization is done to ensure that the network is not ill-conditioned. This helps us issues associated with how common activation functions (like tanh) transform very large (positive or negative) numbers, also, helping us to avoid related issues when computing derivatives. Normalization of series can be done by (1) Standard deviation (2) Scikit function. This project uses scikit function for the series to lie in the range [0,1].

It transforms features by scaling each feature to the given range. Lets' look at the normalized time series we will be making predictions on.

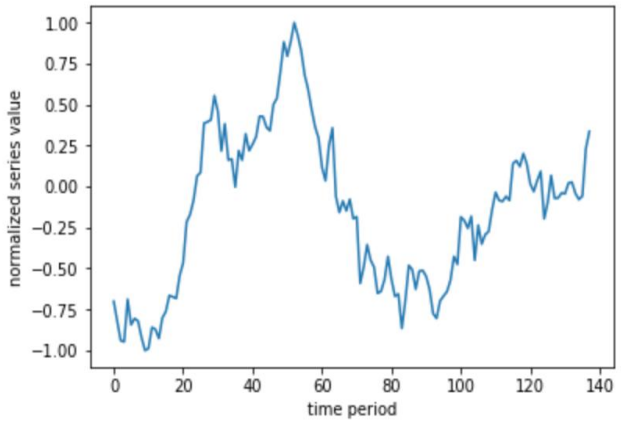

Fig. 1. Plot of normalized time series

IV. PART II: CUTTING THE TIME SERIES INTO SEQUENCES

Our time series is a sequence of numbers represented as n0, n1, n2,……np. In the above sequence, np is the numerical value at the time instant 'p'. We consider P to be the total length of the series. In order to apply our RNN we treat the time series prediction problem as a regression problem. We need to use a sliding window to construct a set of associated input/output pairs to regress on. For example, if we use a window size of T=5, following would be the input/output pairs that would get: -

TABLE I. Input-Output pairs when T=5

INPUT	OUTPUT
{ n1, n2, n3, n4, n5 }	n6
{ n2, n3, n4, n5, n6 }	n7
…….	……
{ np-5, np-4, np-3, np-2, np-1 }	np

We implement a function that runs a sliding window. The inputs given are i) series, ii) window length and it is expected to return the input/output sequences. We can test our function on a list of odd numbers**. [1, 3, 5, 7, 9, 11, 13]**

```
--- the input X will look like ----
[[ 1  3]
 [ 3  5]
 [ 5  7]
 [ 7  9]
 [ 9 11]]
--- the associated output y will look like ----
[[ 5]
 [ 7]
 [ 9]
 [11]
 [13]]
```

Fig. 2. Output after running a window of size 2

V. SPLITTING INTO TRAINING AND TESTING SETS

We segregate the entire database into training and testing sets. We will pare down the last one-third of it for validation. Just like any regression problem, we need to have something to test on after we train the model.

Splitting is not done randomly because of the temporal relationship between input and output pairs. Random splitting poses the risk of receiving new points with the timeframe of our training set. We thus, set a split point using ceil function.
Training set -> 2/3 Testing set -> 1/3
Once the sets are ready we could build the RNN model.

VI. BUILD AND RUN A REGRESSION MODEL

To use Keras's RNN LSTM module, input must be reshaped to [samples, window_size, stepwise].

Keras is a popular deep learning framework which offers consistent and simple APIs, minimizing the number of user actions required for common use cases. Keras being an open source neural network library written in python, is capable of running on TensorFlow, Microsoft Cognitive Toolkit or Theano. Using Keras, we build a two hidden layer RNN of the certain specifications.

An optimizer is one of the two arguments required for compiling a Keras model. We can either instantiate an optimizer or call it by the function name. As discussed earlier, we used Long Short Term Memory layer.

The specifications are as follows: -

• Layer 1 uses an LSTM module with 5 hidden units having the activation function tanh.
• Layer 2 uses a fully connected model with one unit.
•Since we are performing Regression here, 'Mean_Squared_error' must be used.

The reshaped input in the form of [samples, window_size, step_size] is [88, 7, 1].

To run the model, number of epochs is fixed to be 1000 and we chose the batch size of 50. The verbose parameter was toggled monitor the progress of the algorithm. The plot of number of epochs V/s loss is given below.

In order to check the model performance, we have performed predictions on both training and testing sets.If the training error is greater than 0.02, we need to re-train the model by using a higher epoch value.

LSTM stands for Long short-term architecture,

The plot of losses in history is given below. The plot is number of epochs V/S loss.

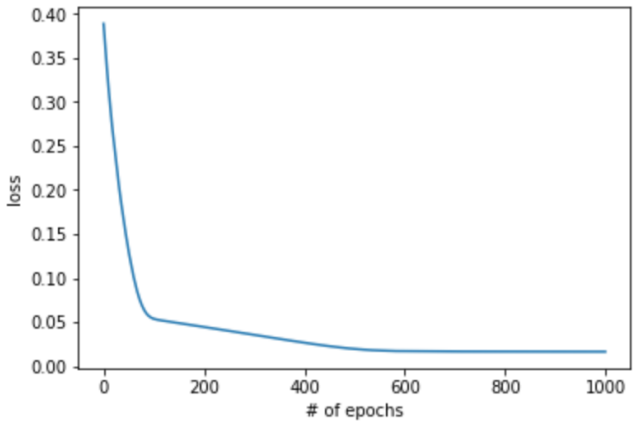

Fig. 3. The plot of losses in history

VII. RESULT AND DISCUSSION

After generating the predictions, we have printed the training and testing errors. The training and testing errors have been found as follows: -

```
training error = 0.016313680363
testing error = 0.0139049960214
```

Fig. 4. Training and Testing error

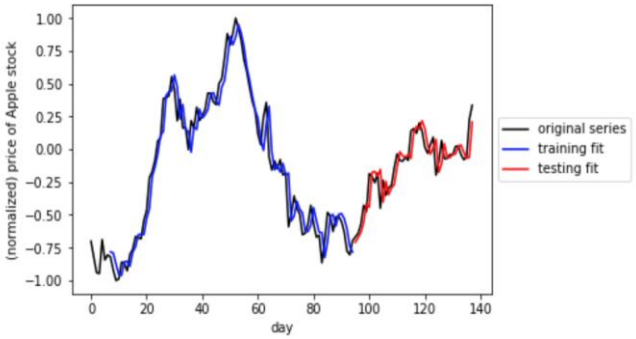

Fig. 5. Plot of original series as well as predictions

VIII. CONCLUSION

With just a deviation of 1.6% with respect to training set and 1.3% with respect to testing error, we have successfully predicted the stock prices of the particular dataset. This model of RNN can also be utilized for text series prediction and market trends of various other companies. Long Short Term Memory Architecture can be used to accurately to predict Stock prices. The future scope of this algorithm is to use the model for Automatic Text Generation or any other Time series Prediction.

REFERENCES

1. Chang S V, Gan K S, On C K, et al. A review of stock market prediction with Artificial neural network (ANN). IEEE International Conference on Control System, Computing and Engineering. IEEE, 2014:477-482.
2. Yakub M U , Al-Ahmadi M.S Application of combined Arma-Neural Network Models to predict Stock Prices. The Multidisciplinary International Social Networks Conference on Socialinformatics 2016, Data Science. ACM, 2016:40.
3. Jabin S. Stock Market Prediction using Feed-forward Artificial Neural Network. International Journal of Computer Applications, 2014, 99(9):4-8.
4. Jain L C, Medsker L R. Recurrent Neural Networks: Design and Applications. Joint Conference on Neural Networks. 1999:1537 - 1541.
5. Gers F A, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Computation, 2000, 12(10):2451.
6. Jia H. Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction. 2016.
7. Chen K, Zhou Y, Dai F. A LSTM-based method for stock returns prediction: A case study of China stock market. IEEE International Conference on Big Data. IEEE, 2015:2823-2824.
8. Xie X K, Wang H. Recurrent neural network for forecasting stock market trend. Computer Science, Technology and Application: Proceedings of the 2016 International Conference on Computer Science, Technology and Application (CSTA2016). World Scientific, 2016: 397.
9. Rather A M, Agarwal A, Sastry V N. Recurrent neural network and a hybrid model for prediction of stock returns. Expert Systems with Applications, 2015, 42(6):3234-3241.
10. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Aistats. 2010, 9: 249-256.
11. Le Q V, Jaitly N, Hinton G E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. Computer Science, 2015.
12. Kingma D, Ba J. Adam: A Method for Stochastic Optimization.

AUTHORS PROFILE

Dr. Middi Appala Raju is a professor at Christ (Deemed to be university) specializing in Financial Management. He obtained his phD from Andhra University in Banking and Financial Services. His research interests include Data Analytics, AI in Finance, Stock Market Analysis etc.

Venkata Sai Rishita Middi is currently pursuing her Masters from Brown university, USA in Electrical and Computer Engineering. Her Research interests include Computer Vision, Natural Language processing and Neural Networks.