Recurrent Neural Network For Time-Series Prediction of Stock Prices

[1]Asif Sohail Mohammed
Department of Computer Science
University of Texas at Dallas
Richardson, USA
Email: axm190041@utdallas.edu

[2]Aikansh Priyam
Department of Computer Science
University of Texas at Dallas
Richardson, USA
Email: axp190019@utdallas.edu

Abstract *–* ***With the rise of Artificial Intelligence and Machine Learning people have been trying to use it for predicting stock prices so that they will have better chances of profit. The Machine learning algorithm used in this project is the Recurrent Neural Network. LSTM which is an implementation of RNN has been implemented in this project. The close prices of the stocks were used as input and the close price of the stocks was predicted.***

Keywords – Stock prediction, Recurrent Neural Network, LSTM

1. Introduction

Over the past few years, the stock prediction has been very crucial for quantitative analysts and investment companies. There are many data available for the stocks of various companies, for example open, close, low, high prices, and the volume of stocks sold on a particular day. The price of stocks is influenced by many factors for example quarterly profit or loss of the company, downfall of the market, change in market trends of the company, and many others.

There are some industries like banking sectors and financial services that use a huge amount of stock trend information to analyze the stock price. Before the use of Artificial Intelligence and Machine Learning, the investors would use some technical analysis, for example, charts of companies, news articles from newspapers and magazines, and market indices. But this analysis not always gives correct results for the future stock price. There have been many Artificial Intelligence approaches used for stock prediction. For instance, financial news was used to identify the sentiments of the news to predict whether it will be useful or not to buy the shares of a company. Sentiment analysis is a good way to understand an opinion about a given subject from the news articles. Few other machine learning techniques have also been used for stock prediction like Support vector machine, Artificial Neural network, Least Square- Support Vector Machines. SVM was used in stock market prediction in [1]. Support vector machine is a statistical machine learning that is applied in both regression and classification problems. SVM model learns by minimizing the risk function and the empirical error and regularization terms have been derived to minimize the structural risk [2]. Box et al. presented a revised least squares (LS)-SVM model and predicted movements in the Nasdaq Index after training with satisfactory results [3].

In our project, we have used a deep learning model, which is an extension of the Artificial Neural network. Many studies have used to predict financial time series. For example, Ting et al. used a deep convolutional neural network to forecast the effect of events on stock price movements [4]. Bengio et al. used long-short term memory (LSTM) to predict stock prices [5]. A time-series data can be thought of as waveform data. Bao et al. used a model that combines wavelet transform and stacked autoencoder (SAE) to de-noise a financial time series [6].

Our project used a dataset of S&P 500 from Kaggle for creating the model and predicting the test data. The dataset contains a CSV file which contains open, close, high, and low price for a particular date. We used close prices of the dataset of ALXN and predicted the closing prices of the stock. The dataset was split between train and test data in the ratio of 80:20. LSTM which is an implementation of Recurrent Neural Network has been used to create a model. This model was used to predict test data and the predicted test data was plotted in the same graph with the real values of the data.

The rest of the project report is organized as follows. Section 2 of the report contains the Theoretical and conceptual study of the technique/algorithm used in this project. Section 3 of the report contains results and analysis. Section 4 of the report contains a conclusion and future work. Section 5 is the final section and it contains the references used for working on the project.

2. Theoretical and conceptual study of the algorithm

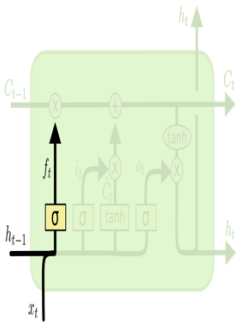
In our project, we have used a Recurrent Neural Network to predict time series stock. We have implemented Long short-term memory (LSTM) which is an implementation of RNN. Recurrent Neural Networks are a class of Artificial Neural Networks. RNNs have numerous applications like neural machine translation, document summarization, predicting time series, speech recognition. In a RNN connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. There are two common variations of RNN – Bidirectional RNN and Depp RNN. There are many advantages of using RNNs, for example, It can process the input of any length, model size does not increase with an increase in the size of the input, weights that are assigned to the networks are shared across time.

In our project, we have implemented LSTM which is a different kind of RNN in which the previous input can be remembered by the network for a longer time and improves the accuracy of the model when the context needed by the model is more. When the gap increases between the words LSTM performs better than RNN.

LSTMs helps to avoid long term dependency problem. In standard RNNs, the repeating modules have a simple structure. LSTMs have chain-like structures, but the repeating module has a different structure. Unlike RNN that has one neural network layer, LSTMs have four different layers that interact in a special way.

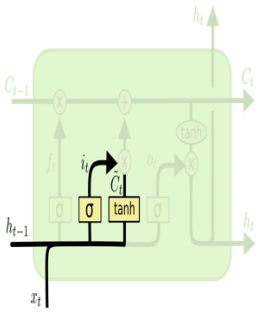
The key to LSTMs is the cell state. The cell state is similar to a conveyor belt that runs straight down the entire chain. Because of this feature, the data can flow easily without losing any data. LSTMs can add or remove data to the cell state which is structured by the gates. Gates are used to optionally let some of the data which is important for the model. Gates are created using the sigmoid neural network layer and a pointwise multiplication operation. The sigmoid layer which is used for the gates gives a scaled output between 0 and 1. 0 means no data should be allowed to go through and 1 means all the data should be
allowed to go through and 1 means all the data shallowed to pass through the gates. LSTM has three gates, input gate, forget gate, and output gate and these are the gates that are used to protect and control the cell state.

The first step in LSTM it to decide what data needs to be sent to the gates and what needs to removed. Forget gate layer of the LSTM is used to decide which data needs to be removed. Forget layer looks at h_{t-1} and x_t and gives an output between 0 and 1 for each number in the cell state C_{t-1}.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

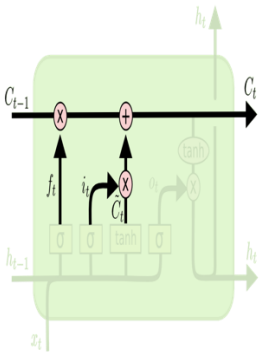
After the forget layer decides what information to pass, LSTM decides what to do with new data. The input gate layer decides which information needs to be updated. After passing the data through the input layer, a tanh layer is used to create a vector of new candidate values, $\tilde{C}$ tC~t, that could be added to the state. After this, two outputs are combined to update the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

The old cell state, C_{t-1} is updated to cell state C_t. The old state is multiplied by f_t, and the data that was decided to forget will be forgotten. Then $i*C_t$ is added.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3. Training, Results, and Analysis

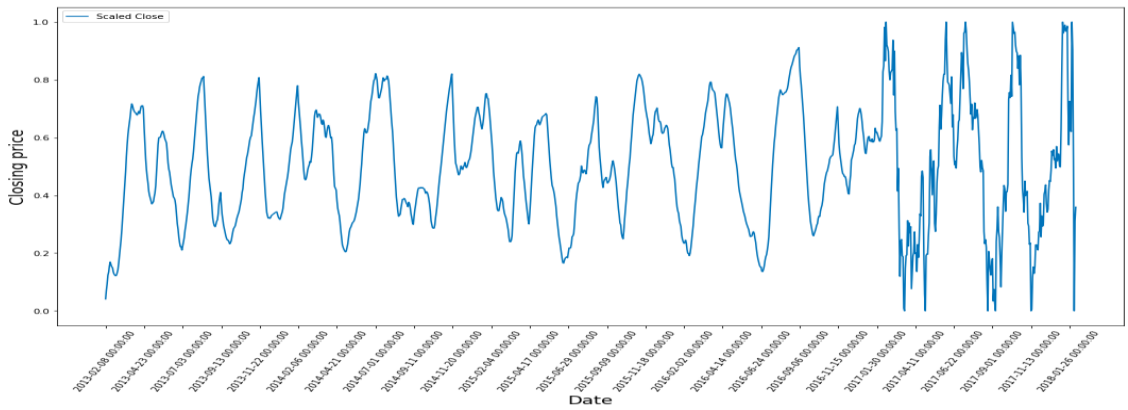
The datasets that were used in our project were S&P 500 from Kaggle which contains stock prices of various companies between 2013 and 2018. The dataset contains open, close, high, and low price volume of stocks traded on a particular day. We used the closing prices of the dataset to predict the future closing price of the stocks. The dataset was divided into training and test datasets in the ratio of 80:20. The close price of the American Airlines company was taken, and the data was pre-processed using the scikit-learn library. The close prices were normalized so that its value ranged between 0 and 1. The normalized data has also been plotted in the graph.

After pre-processing the data, a window size of 50 was taken and input data was created. For every training data of 50 samples, 1 output data was created. We were using 50 data as samples and with the 51st data were checking if we can predict the price correctly or not. After creating the X and Y input data it was distributed between X_train, Y_train, and X_test, Y_test in the ratio of 80:20. Three-step implementation of LSTM has been explained in the image below.

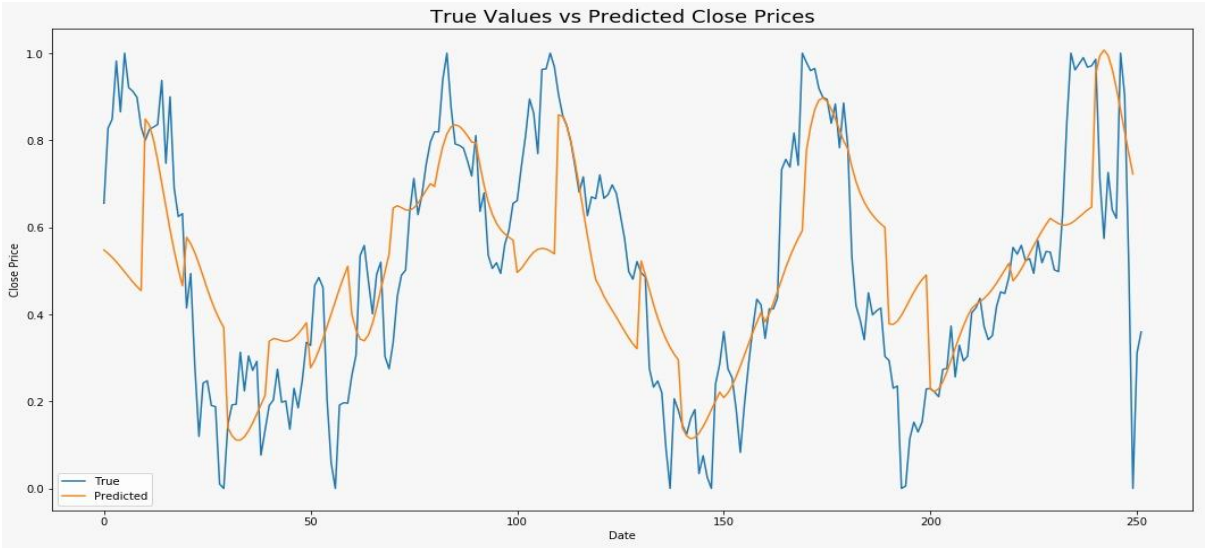
We started implementing our LSTM with 3 hidden layers with 256,256,128 neurons in each layer. After that, we implemented data for 4 hidden layers with 256,256,128,128 neurons in each layer. We changed the number of layers and the number of neurons in the layer and created our model. A table has been made to demonstrate how our output varied with respect to changing parameters.

Table I: Result Log

Exp No.	Parameters Chosen	Results
1	Epochs = 15 Batch Size = 25 Nodes Hidden Layer = [256, 256, 128, 128] Hidden Layers = 4 Dropout Probability = 0.20 Optimizer = Adam Error Function: MSE	Train/Test Error: 80:20 Size of Dataset: 1259 Test Mean Absolute Error = 0.15962031 Test Mean Squared Error = 0.020324983 Test Mean Squared Log Error = 0.018992912
2	Epochs = 15 Batch Size = 75 Nodes Hidden Layer = [256, 256, 128, 128] Hidden Layers = 4 Dropout Probability = 0.20 Optimizer = Adam Error Function: MSE	Train/Test Error: 80:20 Size of Dataset: 1259 Test Mean Absolute Error: 0.16293542 Test Mean Squared Error: 0.022401173 Test Mean Squared Log Error: 0.019874964
3.	Epochs = 15 Batch Size = 20 Nodes Hidden Layer = [256, 128, 128] Hidden Layers = 3 Dropout Probability = 0.20 Optimizer = Adam Error Function: MSE	Train/Test Error: 80:20 Size of Dataset: 1259 Test Mean Absolute Error: 0.15581408 Test Mean Squared Error: 0.020626638 Test Mean Squared Log Error: 0.018587502
4.	Epochs = 15 Batch Size = 20 Nodes Hidden Layer = [256, 256, 128] Hidden Layers = 3 Dropout Probability = 0.25 Optimizer = Adam Error Function: MSE	Train/Test Error: 80:20 Size of Dataset: 1259 Test Mean Absolute Error: 0.15591307 Test Mean Squared Error: 0.02030694 Test Mean Squared Log Error: 0.01852634



Normalized data of the closing price of American Airlines

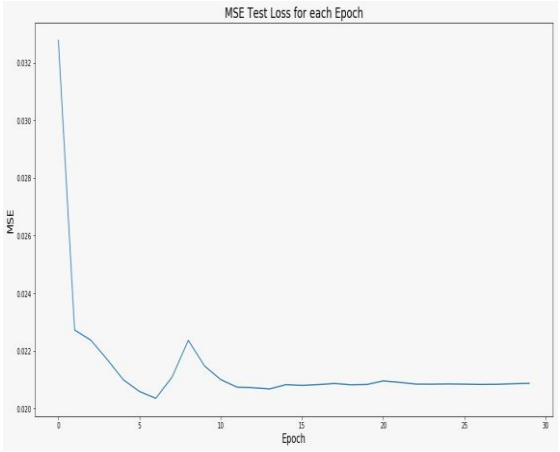


A plot between real and predicted testing data when no. of layers = 4 (256,256,128,128 ->neurons)

4. Conclusion and Future Work

In our project, we used a deep learning network, Long term short memory which is a special kind of Recurrent Neural Network in which context can be read saved for a longer time and improve the model accuracy. We first pre-processed the dataset and normalized the data scaling between 0 and 1. LSTM models were created with a different number of hidden layers and a different number of neurons in each layer. We found better results when the number of hidden layers were 3 and the number of neurons in each layer was 256, 256, 128. As can be seen in the above graph the MSE test loss becomes almost constant after 14-15 epochs, we ran our code till 15 epochs.

Our future work will be on improving the model by changing the parameters so that the stock prices will become more accurate.



MSE wrt number of epochs

5. References

[1] Chun C, Qinghua M, Shuqiang L.: "Research on Support ©Springer, Advances in Intelligent and Soft Computing Volume 148, pp 607-612, 2012.

[2] J. Chen, "Svm application of financial time series forecasting using empirical technical indicators," in Information Networking and Automation (ICINA), 2010 International Conference on, vol. 1. IEEE, 2010, pp. V1–77.

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control. John Wiley & Sons, 2015.

[4] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction." in Ijcai, 2015, pp. 2327–2333.

[5] Y. Baek and H. Y. Kim, "Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module," Expert Systems with Applications, vol. 113, pp. 457–480, 2018.

[6] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," PLOS ONE, vol. 12, no. 7, p. e0180944, 2017.