Recurrent Neural Network for Time Series Prediction on Stock Market Data using LSTM

Name: Saketh Reddy Basika
Department: Computer Science
Institution: The University of Texas at Dallas
City, Country: Richardson, United States
Email: sxb190022@utdallas.edu
NET ID: SXB190022

Name: Sandeep Reddy Gopu
Department: Computer Science
Institution: The University of Texas at Dallas
City, Country: Richardson, United States
Email: sxg180156@utdallas.edu
NET ID: SXG180156

Abstract— **Foreseeing stock market is one of the most troublesome errands in the field of computation. There are numerous elements engaged with the expectation – physical variables versus physiological, financial specialist estimation, showcase rumors. Every one of these perspectives join to make stock costs unpredictable and hard to foresee with a high level of precision. We examine data analysis as a distinct advantage in this domain. As per effective market hypothesis when all data identified with an organization and financial exchange events are in a split second accessible to all partners/stakeholder's, at that point the impacts of those occasions as of now implant themselves in the stock price. In this way, it is said that just the recorded spot value conveys the effect of all other market occasions and can be utilized to anticipate its future development. Subsequently, thinking about the past stock cost as the last sign of all affecting components we utilize Machine Learning (ML) methods on verifiable stock value information to derive future pattern. ML strategies can possibly uncover examples and bits of knowledge we didn't see previously, and these can be utilized to make unerringly precise forecasts. We propose a system utilizing LSTM (Long Short-term Memory) algorithm and organization's stock data development a model to predict future stock prices.**

I. Introduction

We humans have the ability to understand patterns which gives us the ability to predict and analyze future actions. Recurrent Neural Network is somewhat similar to the above discussed concept. In order to get the understanding of Neural Networks we first need to discuss what is meant by Deep Learning. Deep Learning deals with the algorithms that replicate the structure of human brain cells which are called as 'Artificial Neural Networks'. RNN is a type of artificial neural net which are used to predict sequential data such as texts, numerical time series data. RNN is basically combination of 2 or more perceptron's. More the number of hidden layers, the developed model will be able to predict data with more accuracy. The major drawback of Recurrent Neural Network is vanishing gradient, which means the developed model will face issues in learning long-range dependencies. In this project we have stock data of Google from the year 1970 to 2019. We need to develop a model to predict the data 1970 May to 2019 November. Due to the drawback RNN model would not predict the data accurately. To solve this issue, we need to implement Long Short-Term Memory Cell (LSTM) which is a type of RNN.

II. Theoratical and conceptual study

A. Understanding Recurrent Neural Network:

RNN is a type of artificial neural network which make use of simple perceptron with multiple hidden layers. Its used to learn pattern from sequential data and predict future values.

RNN Structure:

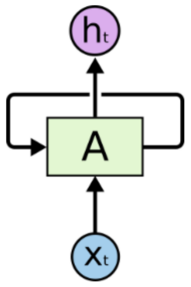


Fig1. Simple Recurrent Neural Network

Fig1 is a representation of simple RNN where A is a neural network, Xt is the input and ht is the output of the network. So the output ht produced by this network is compared with the test data. Then we will get the error rate and now we will do back propagation so that we can adjust the weights based on the error rate to yield better results.

B. What is LSTM?

LSTM is a popular type of Recurrent Neural Network which is used to overcome the drawback of RNN and to learn long range dependencies. LSTM was proposed in 1997 by Sepp Hoch Reiter and Jurgen Schmid Huber and improved in 2000 by Felix Gers' team. LSTM models are capable of remembering information for long time. LSTM can determine how long to keep track of old information, when to remember and forget old data and how to make use of old data to predict new inputs.

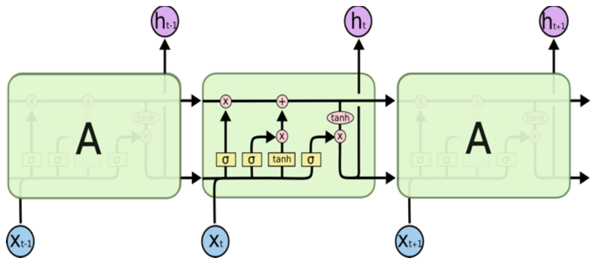


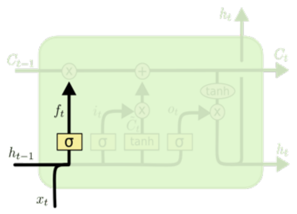
Fig2. LSTM Network

LSTM has a chain of modules similar to Recurrent Neural Network but in RNN there is a chain of repeating modules has a simple structure such as single sigmoid layer and in LSTM the repeating modules has different structure such as having 4 layers. The major part of Long Short-Term Memory is the cell state. LSTM has the ability to add or remove information from cell states by making use of gates. LSTM consists of three gates to control the cell states.

C. LSTM Gate Layers:

LSTM weights are determined by operation gates such as forgot gate, input gate, and output gate. The first step is to make a decision on what information is to be removed and this decision is made by the forgot gate layer.

1. **Forgot Gate:**



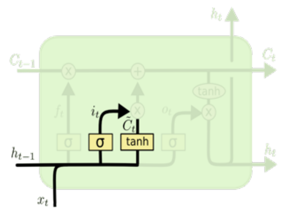
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Ft = sigmoid (Wf [h_t -1, Xt] + bf); where Wi [ht-1, Xt] are weights of hidden and input layers at time t of forgot gate layer and bf is the bias

- This is a sigmoid layer that takes the yield at t-1 and the present contribution at time t and afterward consolidates them into single tensor. It at that point applies straight change pursued by sigmoid. The yield of the door is somewhere in the range of 0 and 1 because of the sigmoid function. This number is then duplicated by the inner states along these lines, it's known as the forgot state.

  If ft = 0, then the previous state is completely forgotten, if ft = 1 the previous state is unaltered.

2. **Input Gate:**



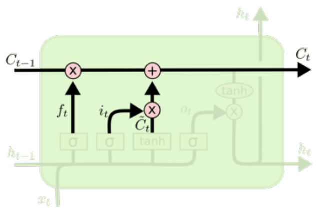
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

it = sigmoid(Wi[ht-1, Xt] + bi)

This state takes the past yeild together with the new inputs and goes them through another sigmoid layer. This gate restores a value somewhere in the range 0 and 1. The value of the input gate is then increased with the yeild of the competitor layer.

Ct=tanh(Wc[ht-1,Xt]+bi)

This layer applies hyperbolic digression to the blend of the input and past yield, restoring the competitor vector. The competitor vector is then added to the internal state, which is refreshed with this standard:

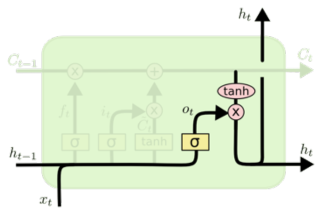


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Ct=ft *Ct-1+it*Ct

The past state is increased by the forgot gate, and afterward added to the division of the new competitor permitted by the output gate.

3. **Output Gate:**



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ +\right.$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Ot=σ(Wo[ht-1,Xt]+bo)

ht=Ot*tanh Ct

This gate controls the amount of the internal state is passed to the yield and works along these lines to other gates.

D. *What is Time Series Analysis:*

Time series analysis is a method for analyzing time series or historical data and to make predictions and extract future data from it. Time series includes the utilization of information that are indexed by equally spaced increments of time. Because of the discrete idea of time series data, many of it's indexes have an occasional as well as pattern component incorporated with the information. The initial phase in time series modeling is to represent existing seasons as well as patterns.

E. *Implementaion:*

1. **Importing Libraries:**
   We start by importing basic libraries for data loading and data preprocessing along with tensorflow library.
2. **Loading the dataset:**
   The stock_data.csv file contains the prices and volume of Google stocks. We load this file into our working environment using pd.read_csv() command
3. **Deciding on a target variable:**
   We select the 'Closing Price' as our target variable.
4. **Data preprocessing:**
   We are going to scale our data before fitting it in our model.
5. **Labeling the dataset:**
   We create features and labels to our dataset by windowing the data.
6. **Dividing the given data into training and testing data:**
   We have used eighty percent of the data from the dataset for training and twenty percent of data from the dataset for testing.
7. **Defining the network:**
   Here, we are going to define the hyper parameters such as batch_size, window_size, hidden_layers, clip_margin, learning_rate, epochs
8. **Defining the placeholders:**
   They allow us to send different data within our network with the tf.placeholder() command.
9. **Defining LSTM weights:**

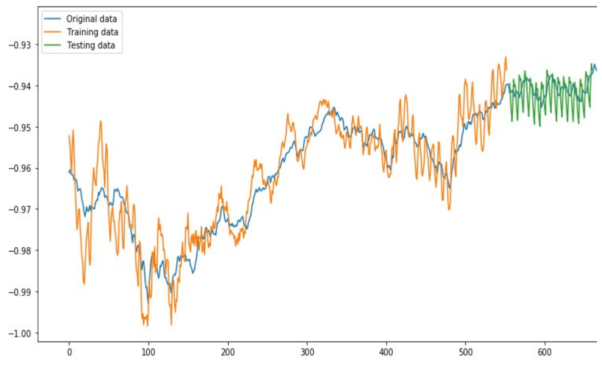
LSTM weights are determined by Operation Gates which includes forget, input and output gates.

10. **Defining the loss:**
    Here we will use mean_squared_error() command for the loss of minimize the errors.
11. **Training the network:**
    We train our model bases on the epochs we initialized and observe change in loss. We can see that the current loss will be decreased as the number of epochs increases which in turn increases the prediction accuracy.
12. **Plotting the predictions:**
    Now after the model is developed and the output is generated, we plot our output on a graph.

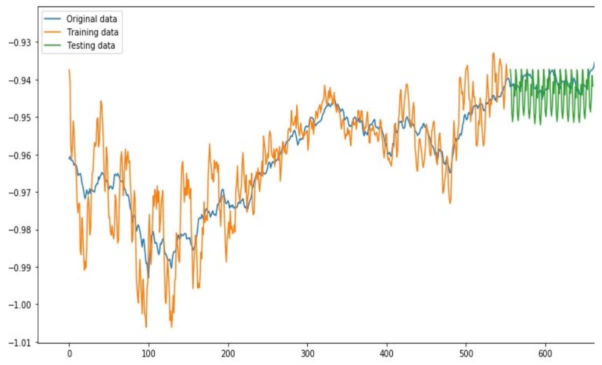
III. Results and Analysis

Graphical representation of original, trained, prediction data by considering different values of epochs

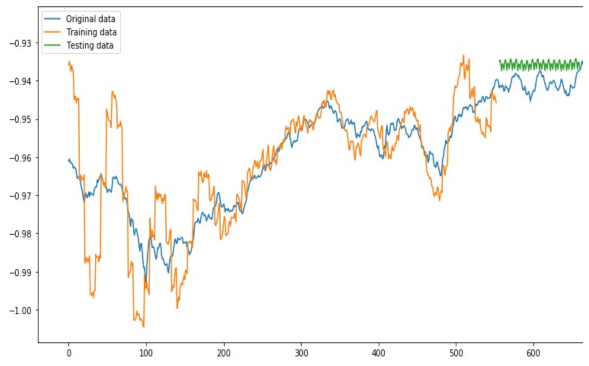
1. Epoch = 25



2. Epoch = 50



3. Epoch = 100



IV. Conclusion and Future Work

LSTMs were a major advance in what we can achieve with RNNs. It's normal to ponder: is there another enormous advance? A typical conclusion among analysts is: "Yes! There is a subsequent stage and it's consideration!" The thought is to give each progression of a RNN a chance to pick data to take a look at from some bigger collection of data. For instance, in the event that you are utilizing a RNN to make a subtitle depicting a picture, it may pick a piece of the picture to take a look at for each word it yields. Truth be told, it may be a fun beginning stage on the off chance that you need to investigate consideration! There's been various truly energizing outcomes utilizing consideration, and it appears as though much more are around the corner.

In this project we tend to predict the share values of a company based on the information we had. The next thing that we can do is, we can take share data of various companies from different sectors which has various share patterns so that we can study the share patterns of different sectors and analyze a graph with even more accuracy.

References

[1] M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using LSTM," International Journal of Science and Research (IJSR), vol. 6, no. 4, pp. 1754-1756, 2017.

[2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in

International Conference on Advances in Computing, Communications and Informatics, 2017.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[4] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 6, no. 2, pp. 107-116, 1998.