Stock Price Trends Prediction Using LSTM Neural Networks

Ganesh Anirudh Panthula
Department of Electrical and Computer Engineering
Florida State University
Tallahassee, Florida
agp16b@my.fsu.edu
Prof: Dr. Simon Foo

***Abstract*—Predicting Stock Prices has been a challenge for many years due to the fact that it is extremely complex and hard to find any one method which can be applied for consistent results. This article uses a special kind of Recurrent Neural Networks called Long Short Term Memory to use data in the form time series to predict trends in the stock prices.**

I. Introduction

Stock market prediction have been an object of studies for many years, but given its high complexity, no of variables and sources included it has been proven to be a very difficult task. With the increase in use of machine learning algorithms and techniques in a wide variety of disciplines, stock market prediction has also been improved using different machine learning techniques.

This paper explains a way of finding trends in stock market prices using historical data. Recurrent neural networks (RNNs) is found to be one of the powerful models for processing sequential data such as time series data [1], while Rnn's are good at learning short term dependencies, it can't be used for long term dependencies as interactions between time steps that are several steps apart are difficult to keep track with. To overcome this problem a special type of recurrent neural network, long short term memory (LSTM) is used. [2]

Data comprising the closing stock price of S&P 500 companies from 2010 to 2016 will be used as source of information for the network. Prices were taken from Yahoo Finance, fundamentals are from Nasdaq Financials, extended by some fields from EDGAR SEC databases. Upon this dataset, the model will be trained and evaluated to plot a graph which can be used to analyze trends in the closing stock price of any S&P 500 company.

II. Different methods used and related work

There have been numerous papers, articles and blogs posted about the prediction of stock prices, from different disciplines such as economics, physics, computer science and statistics. In 2012, more than 80% of trades in the United States stock market were done by different algorithms. [3] The Efficient Market hypothesis states that the present price of an asset is always reflects all of the previous data associated with it [4], on the contrary The Random walk hypothesis [5] states that stock price is independent of its history, in other words present stock price depends only on the present parameters and has no relation with previous information.

In regards to Different machine learning models, Support Vector Machines (SVM) have been used for statistical learning [6], others methods like Neural networks, Reinforced Learning, Natural Language Processing with textual analysis where in Tweets from Twitter.com have been used to alter weights in Neural networks so that the model produces better results NLTK (Natural Language Tool Kit) is widely used for such methods. For works related to Deep learning in stock prediction there is a study made on Deep Belief Network (DBN) where in Boltzmann Machines are used along with Multi-Level Perceptron to predict above-median returns. [7]

III. Long Short Term memory

In a traditional neural network, there is a forward – propagation phase and a backward-propagation phase. During the backward-propagation phase the weights are updated, but in a few situations the gradient signal is being multiplied a large number of times, as many as the number of time steps by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. This magnitude of weights in the transition matrix can have a huge impact on the correctness of the learning process. [8]

There two possible problems associated with the updating of weights in a recurrent neural network, first is the Vanishing Gradient Problem, where in the weights are so small that it can lead to a situation where in the gradient signal gets so small that the learning either becomes very slow or stops working altogether, this in turn makes the task of learning long term dependencies in the data difficult. The second problem of a recurrent neural network is the Exploding Gradient Problem, here the weights of the matrix are large, in other words the leading eigenvalue of the weight matrix is larger than 1.0, this

leads to a situation where the gradient signal is so large that it can cause learning to diverge.

These issues can be addressed using the LSTM networks. A LSTM model consists of a structure called memory cells, a memory cell is composed of three gates and a neuron with a self-recurrent connection (a connection to itself). The three gates are input gate, output gate and forget gate, these gates serve to modulate the interactions between the memory cell itself and the environment around it. The input gate modulates the incoming information into the memory cell, where in it only allows relevant information or blocks it, on the other hand the output gate can allow the state of memory cell to have an effect on other neurons or prevent it. The self-recurrent connection is modulated by the forget gate, which allows the cell to remember or forget its previous state, as needed.

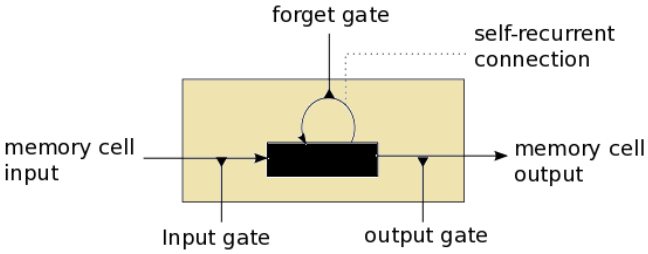

Figure 1: illustration of a LSTM memory cell

The equations below show how a layer of memory cells is updated at every time step t. In these equations: [9]

- x_t is the input to the memory cell layer at time t
- $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and V_o are weight matrices
- b_i, b_f, b_c and b_o are bias vectors

First, we compute the values for i_t, the input gate, and $\widetilde{C_t}$ the candidate value for the states of the memory cells at time t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (2)$$

Second, we compute the value for f_t, the activation of the memory cells' forget gates at time t:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3)$$

Given the value of the input gate activation i_t , the forget gate activation f_t and the candidate state value $\widetilde{C_t}$ we can compute C_t the memory cells' new state at time t:

$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1} \quad (4)$$

With the new state of the memory cells, we can compute the value of their output gates and, subsequently, their outputs:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

IV. Methodology and Development

The API used to build the model is Keras on TensorFlow backend, Amazon Web Services (AWS) was used to reduce the computation time for building the model.

A. Data Processing

Dataset used to build the model is the closing price of S&P 500 from January 2000 – August 2016. This is a series of data points listed in time order or time series. The goal is to predict the closing price of any given date after training.

The dataset is normalized before using it in the model as it improves convergence. Give below code snippet of normalize function is used to normalize percentage changes from the starting point.

```
def normalise_windows(window_data):
    normalised_data = []
    for window in window_data:
        normalised_window = [((float(p) / float(window[0])) - 1) for p in window]
        normalised_data.append(normalised_window)
    return normalised_data
```

Figure 2 Normalize function code

Here each price is divided by initial price minus one, the data is later de-normalized to get a real world number out of it.

B. Implementation

For training the data is arranged in an order of sequence, a sequence length of 22 days in taken and the data is arranged using the formula

$$M = \beta - \alpha \quad (7)$$

Here M is the maximum date, it is calculated by the difference in the latest date from that of the sequence length.

The model uses two LSTM layers of 256 units each and then a Relu activation function followed by a linear activation

function. Dropout of 0.3 and the number of epochs equal to 90 is considered to be the most suitable parameters for this task.
A snippet of the code showing the model is shown in the figure below

```
def build_model(layers):
    d = 0.3
    model = Sequential ()

    model.add(LSTM(256, input_shape=(layers[1], layers[0]), return_sequences=True))
    model.add(Dropout(d))
    model.add(LSTM(256, input_shape=(layers[1], layers[0]), return_sequences=False))
    model.add(Dropout(d))
    model.add(Dense(32,kernel_initializer='uniform', activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform', activation = 'linear'))

    #adam = keras.optimizers.Adam(decay=0.2)

    start=time.time()
    model.compile(loss='mse',optimizer='adam', metrics=['accuracy'])
    print("compilation time: ", time.time()-start)
    return model
```

Figure 3: Model function code

C. Results

The aim of the model is to predict trends in stock price for a company, this is attained by generating graphs of the closing price. The training and test set are split in the ratio of 80 by 20.

The results obtained are given in the table below

	Root Mean Square Error	Mean Square Error
Train Score	0.04	0.00198
Test Score	0.13	0.01802

This model can be used to find graphs of the trends in stock price for any of the S&P 500 companies, here graphs of Yahoo and Google in figure 4 and 5.

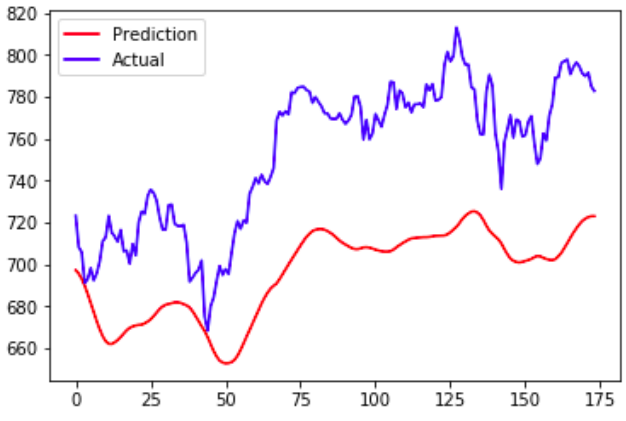

Figure 4 Google stock price

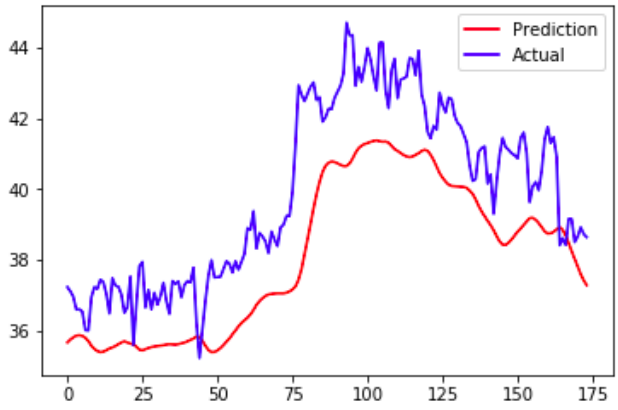

Figure 5 Yahoo stock price

Padding of 20 units is applied to the predicted stock prices as seen in figures 4 and 5, this is so that the trends can be clearly seen as compared with actual stock prices.

V. Conclusions

We can observe that in general this model outperforms the baselines in predicting the trends of the stock price, this outcome can be considered promising as it is able to perform better as compared with other approaches used today.

Although the Training and Test scores can be improved with more data, but the model has demonstrated clear results in terms of predicting trends for any of the S&P 500 companies with the limited data used in building the model.

The graphs developed by the model can be used for different kind of application such as

- Predicting a company's chance of going bankrupt
- Finding an undervalued stock price to invest in
- To find return or investment

In the Forthcoming Research, we intend to improve the model by using different Machine learning techniques like Sentiment Analysis and applying Reinforced learning methods.

References

[1] Saad, Emad W., Danil V. Prokhorov, and Donald C. Wunsch. "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks." Neural Networks, IEEE Transactions on 9.6 (1998): 1456-1470.

[2] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[3] M. Glantz and R. Kissell, Multi-asset Risk Modeling, 1st ed. Academic Press, 2013.

[4] E. F. Fama and B. G. Malkiel, “Efficient capital markets: A review of theory and empirical work,” The Journal of Finance, vol. 25, no. 2, pp. 383–417, 1970. [Online]. Available: http: //dx.doi.org/10.1111/j.1540-6261.1970.tb00518.x

[5] B. G. Malkiel, A Random Walk Down Wall Street. Norton, New York, 1973.

[6] K. Kim, “Financial time series forecasting using support vector machines,” Neurocomputing, vol. 55, no. 1–2, pp. 307 – 319, 2003, support Vector Machines. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231203003722

[7] B. Batres-Estrada, “Deep learning for multivariate financial time series,” 2015.

[8] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166.

[9] "DeepLearning 0.1 documentation," [Online]. Available: http://deeplearning.net/tutorial/lstm.html.