Stock Market Prediction by Recurrent Neural Network on LSTM Model

Nahida Akther
ID: 16155030
&
Md.Masud Rana
ID:16155027
Department of IT,
University of Information Technology and Sciences

Abstract:

The art of forecasting stock prices has been a difficult task for many of the researchers and analysts. In fact, investors are highly interested in the research area of stock price prediction. For a good and successful investment, many investors are keen on knowing the future situation of the stock market. Good and effective prediction systems for stock market help traders, investors, and analyst by providing supportive information like the future direction of the stock market. In this work, we present a recurrent neural network (RNN) and Long Short-Term Memory (LSTM) approach to predict stock market indices.

Keywords: Recurrent Neural Network (RNN), Long short-term memory (LSTM),Matplotlib,Pandas,Numpy,Tensorflow,Sequential, Dense ,Dropouts.

Introduction:

There are a lot of complicated financial indicators and also the fluctuation of the stock market is highly violent. However, as the technology is getting advanced, the opportunity to gain a steady fortune from the stock market is increased and it also helps experts to find out the most informative indicators to make a better prediction. The prediction of the market value is of great importance to help in maximizing the profit of stock option purchase while keeping the risk low. Recurrent neural networks (RNN) have proved one of the most powerful models

for processing sequential data. Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

Recurrent Neural Network (RNN):

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

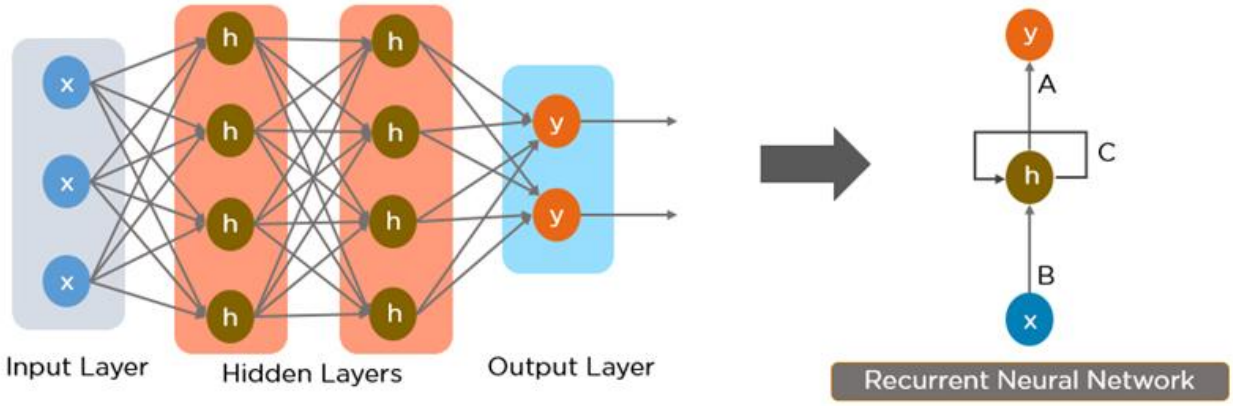

Figure: Recurrent Neural Network (RNN)

Here A, B and C are network parameters, x is the input, h is the hidden stateand y is the output. The expanded version of a RNN looks as follows:

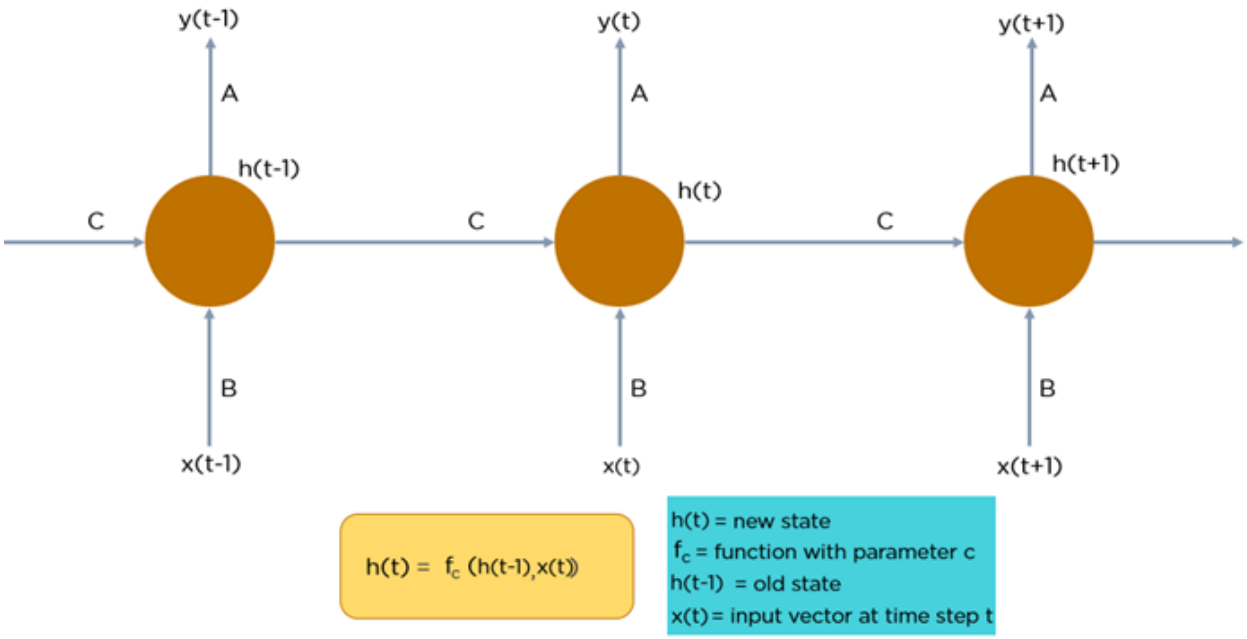

Figure: Example of Recurrent Neural Network (RNN)

Long short-term memory (LSTM):

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

Features we have used:

Technology

- ✓ Anaconda (free open-source)
- ✓ Spyder
- ✓ Python (3)

Methodology:

• **Stage 1**: Raw Data: In this stage, the historical stock data is collected from the Google stock price and this historical data is used for the prediction of future stock prices.

```
dataset =
pd.read_csv('Google_Stock_Price_Train.csv',index_col="Date",parse_dat
es=True)
```

	Open	High	Low	Close	Volume
Date					
2012-01-03	325.25	332.83	324.97	663.59	7,380,500
2012-01-04	331.27	333.87	329.08	666.45	5,749,400
2012-01-05	329.83	330.75	326.89	657.21	6,590,300
2012-01-06	328.34	328.77	323.68	648.24	5,405,900
2012-01-09	322.04	322.29	309.46	620.76	11,688,800

Figure: some of data of Google stock Dataset

• ***Stage 2:*** Data Preprocessing: The pre-processing stage involves: a)Data discretization: Part of data reduction but with particular importance, especially for numerical data b) Data transformation: Normalization. c) Data cleaning: Fill in missing values. d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate and creating a data structure with 60 timesteps and 1 output.

• **Stage 3:** Feature Extraction: In this layer, only the features which are to be fed to the neural network are chosen. We will choose the feature from Date, open, high, low, close, and volume.

```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

Figure: import keras libraries and packages

• **Stage 4:** Training Neural Network: In this stage, the data is fed to the neural network and trained for prediction assigning random biases and weights. Our LSTM model is composed of a sequential input layer followed by 3 LSTM layers and dense layer with activation and then finally a dense output layer with linear activation function**.**

Optimizer:

The type of optimizer used can greatly affect how fast the algorithm converges to the minimum value. Also, it is important that there is some notion of randomness to avoid getting stuck in a local minimum and not reach the global minimum. There are a few great algorithms, but I have chosen to use Adam optimizer. The Adam optimizer combines the perks of two other optimizers: ADAgrad and RMSprop. The ADAgrad optimizer essentially uses a different learning rate for every parameter and every time step. The reasoning behind ADAgrad is that the parameters that are infrequent must have larger learning rates while parameters that are frequent must have smaller learning rates. RMSprop considers fixing the diminishing learning rate by only using a certain number of previous gradients.

The optimizer that are used in this project, and the benefits are summarized into the following:

1) The learning rate is different for every parameter and every iteration.

2) The learning does not diminish as with the ADAgrad.

3) The gradient update uses the moments of the distribution of weights, allowing for a more statistically sound descent.

Regularization:

Another important aspect of training the model is making sure the weights do not get too large and start focusing on one data point, hence overfitting. So we should always include a penalty for large weights. The definition of large would be depending on the type of regularization are used.

Dropouts:

A newer method of preventing overfitting considers what happens when some of the neurons are suddenly not working. This forces the model to not be

overdependent on any groups of neurons, and consider all of them. Dropouts have found their use in making the neurons more robust and hence allowing them to predict the trend without focusing on any one neuron. Here are the results of using dropouts.

• **Stage 5:** Output Generation: In this layer, the output value generated by the output layer of the RNN is compared with the target value. The error or the difference between the target and the obtained output value is minimized by using back propagation algorithm which adjusts the weights and the biases of the network.

```
Epoch 95/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0014
Epoch 96/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0015
Epoch 97/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0017
Epoch 98/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0016
Epoch 99/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0016
Epoch 100/100
1198/1198 [==============================] - 5s 4ms/step - loss: 0.0016
```

Figure: Output Generation

We followed the same for test Data Preprocessing: The pre-processing stage involves a) Data discretization: Part of data reduction but with particular importance, especially for numerical data b) Data transformation: Normalization. c) Data cleaning: Fill in missing values. d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate and creating a data structure with 60 timesteps and 1 output.

Visualization:

All of the analysis above can be implemented with relative “ keras” and their functional API.

```
# Visualising the results
plt.plot(real_stock_price, color = 'red', label = 'Real Google Stock
Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted
Google Stock Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```

Figure: Visualization for result

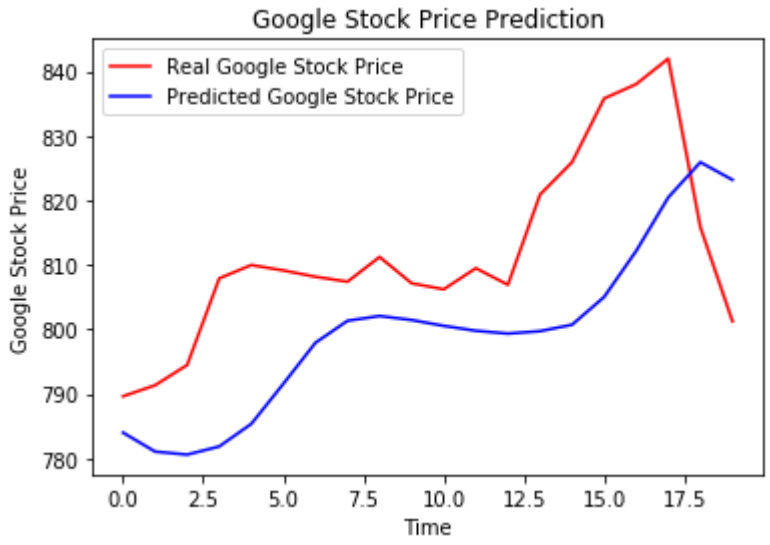

Figure: prediction comparison

Conclusion:

The popularity of stock market trading is growing rapidly, which is encouraging researchers to find out new methods for the prediction using new techniques. The forecasting technique is not only helping the researchers but it also helps investors and any person dealing with the stock market. In order to help predict the stock indices, a forecasting model with good accuracy is required. In this work, we have used one of the most precise forecasting technology using Recurrent Neural Network and Long Short-Term Memory unit which helps investors, analysts or any person interested in investing in the stock market by providing them a good knowledge of the future situation of the stock market.

References:

1) https://en.wikipedia.org/wiki/Long_short-term_memory
2) https://www.quora.com/What-are-recurrent-neural-networks
3) https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/
4) https://skymind.ai/wiki/lstm
5) https://github.com/krishnaik06/Stock-Price-Prediction-using-Keras-and-Recurrent-Neural-Networ/commit/6918aaafb1b11e5087b8424e1a50f4028268a4ea
6) https://github.com/krishnaik06/Stock-Price-Prediction-using-Keras-and-Recurrent-Neural-Networ