Recurrent Neural Networks for Time Series Prediction

Pratik Deepak Naik
Dep. of CS
Univ. Texas Dallas
pxn210006

Puru Jaiswal
Dep. of CS
Univ. Texas Dallas
pxj200018

Akshay Kunnamkalath Mohanakrishnan
Dep. of CS
Univ. Texas Dallas
axk190189

Preema Merlin Dsouza
Dep. of CS
Univ. Texas Dallas
pmd200001

***Abstract*—In Traditional ML algorithms, same priority is given to entire historical data while making predictions. However, in situations like weather prediction or sentence completion, special priority should be given to recently seen data. In this project, we utilize Long short term Memory network to forecast Tesla Stock prices. To do this, we have used python libraries like pandas, numpy, matplotlib and seaborn.**

***Index Terms*—Neural network, LSTM, Neural Networks**

I. Introduction

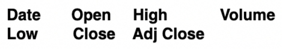
Stock market can be defined as a marketplace for buying and selling stocks which represent a partial ownership of a company. This is facilitated by the Stock exchange who acts as a mediator in this process.
Machine learning is used in a wide variety of fields which include Stock price prediction. This helps to predict the future value of a company and if done correctly, can lead to significant financial gain. There are a lot of factors involved in stock prediction like market behavior, economic and political conditions etc. As a result, it is a very difficult task.
Because of this, we have decided to work on Stock Price prediction for Tesla (TSLA) as part of our main course project. Tesla stock is among the more popular options among traders. Currently valued at around 180$, TSLA hit an all-time high closing price of $409.97 on November 04, 2021.

We have utilized an RNN based architecture as it has a track record of being effective with these classes of problems.
For building our model we have used LSTM(which is RNN-based). This is because LSTM is able to circumvent the issue of vanishing gradients which can happen with other RNN based models.

II. Dataset

A. Description

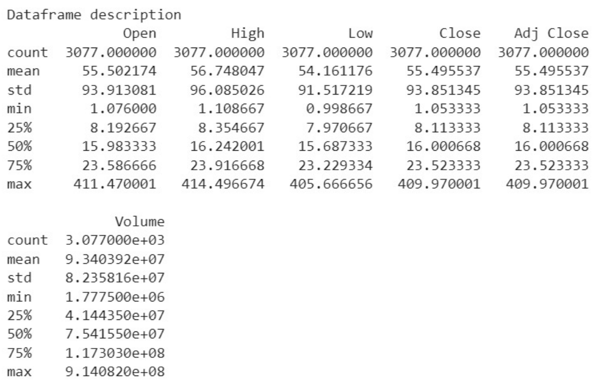
We have used the Tesla Stock price Dataset to build our model. This has 7 features which are :



Date	**Open**	**High**	**Volume**
Low	**Close**	**Adj Close**	

There are 3076 instances in the dataset. The features Open, high, low and close can be used to study the momentum and volatility of the stock.
The momentum of a stock is defined as the speed at which the price is changing. If the open and close values are far apart, then momentum is strong and otherwise, momentum is weak. This is used to understand stock price movements.
Volatility is the amount that a stock's price fluctuates over a specific time period.. If the volatility is high, it represents a higher risk.
Volume represents the number of shares traded for a certain stock over a specific period. High volume stocks have many investors who are interested in buying or selling.
Date attribute is a string object, which is split into Day, Month and Year respectively for analysis purposes. Most companies generate quarterly reports, which is once in 3 Months. The stock price depends on this report; hence we can see a variation in the stock price during each quarter. Therefore, we are adding another attribute called **quarter** to our dataset to help with our analysis.

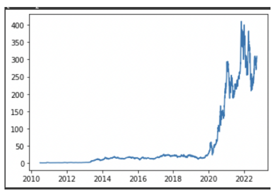
B. Analysis



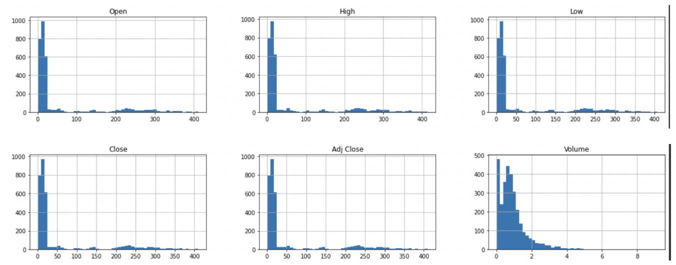
```
Dataframe description
             Open         High          Low        Close    Adj Close
count  3077.000000  3077.000000  3077.000000  3077.000000  3077.000000
mean     55.502174    56.748047    54.161176    55.495537    55.495537
std      93.913081    96.085026    91.517219    93.851345    93.851345
min       1.076000     1.108667     0.998667     1.053333     1.053333
25%       8.192667     8.354667     7.970667     8.113333     8.113333
50%      15.983333    16.242001    15.687333    16.000668    16.000668
75%      23.586666    23.916668    23.229334    23.523333    23.523333
max     411.470001   414.496674   405.666656   409.970001   409.970001

             Volume
count  3.077000e+03
mean   9.340392e+07
std    8.235816e+07
min    1.777500e+06
25%    4.144350e+07
50%    7.541550e+07
75%    1.173030e+08
max    9.140820e+08
```

Overall Summary of the dataset



Line graph for Close vs Year



Histograms representing distribution of the dataset with respect to select features.

The histograms represent the distribution of the dataset with regards to the specific features like Open, High, Low, Close etc. The line graph represents the growth of the 'Close' feature over the years. From these images, it can be observed that the stock price values were lower during the initial years and have experienced a surge over the more recent years.

III. Theoretical study of the algorithm

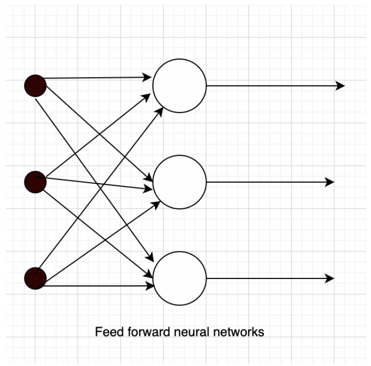
A. Recurrent Neural Networks

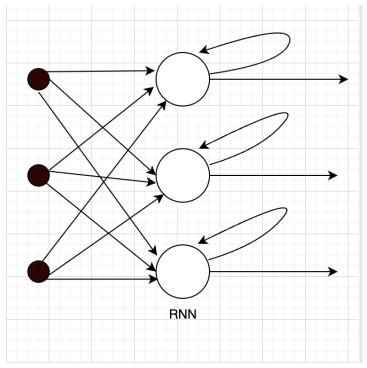
RNNs are considered to be powerful tool because they can be be taught to retain historical knowledge. Even though they were developed in the mid 20th century, their true potential was explored more recently with the abundance of data and rise of computational capabilities.
Based on the historical data stored in their internal memory, RNNs make predictions about the future. Because of this feature, they are favored in scenarios like time series prediction, NLP sentence completion, image captioning.

RNNs are similar to the traditional feedforward and convolutional neural networks in the sense that they utilize training data in order to learn. However, unlike the conventional networks that have no internal memory and therefore no notion of time, the RNNs have internal memory which means that the past ouputs have some weightage in the prediction of the future output.

Another factor that differentiates recurrent networks is that across all the layers of the network, the same parameters are used.In case of feedforward networks, there may be different weights across nodes but in RNN the same weight parameter is used by all of the layers of the neural net. However, using some common neural network techniques like gradient descent and also backpropogation, we stabilize these weights.

Recurrent networks use a technique where it backpropogates through time to calculate gradient This method is a variant of the traditional backpropagation and is specific to sequential data. It is similar to traditional BP in the aspect that by computing errors from output layer to input layer, the model trains itself.However, it is different in the sense that it sums errors at each time step. Feed forward networks don't do this as they don't have the same params across layers.





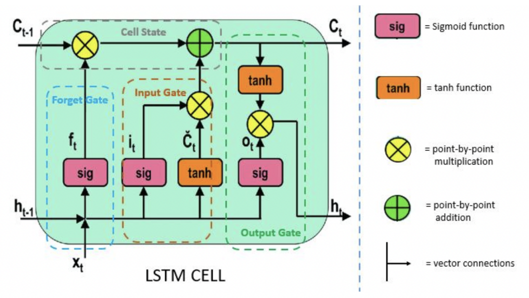
While using RNN for time series predictions, there are two aspects that need to be overcome : Exploding Gradient and Vanishing Gradient. Exploding gradient refers to a scenario where a higher priority is given to a specific neuron and as a result it influences the final result more than the other neuron values. In case of vanishing gradients, the algorithm becomes too small and learning stops or takes a lot of time. This problem is tackled by using the Long short term memort(LSTM).

B. LSTM

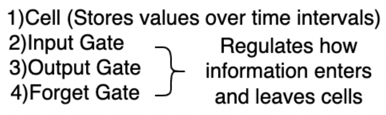
LSTM was first attributed to in the paper **Long Short term memory** by Hochreiter, Sepp, and Jurgen Schmidhuber. Around the year of 2007, it began to transform voice recognition, doing better than conventional models in some speech applications.

LSTM additionally enhanced text-to-voice synthesis and large-vocabulary speech recognition, which was attributed with significantly enhancing Google's speech recognition system's performance.

An LSTM cell has 3 gates, i/p, o/p and forget gate that allows it the capability to remember important things for extended periods of time. Other applications of LSTM include Speech recognition, Robot control, Music composition, Sign language translation, Object co segmentation, Drug design and Human Action Recognition.

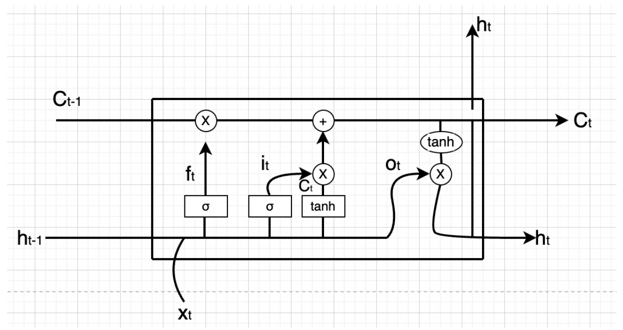


A unit of LSTM consists of the following as highlighted in the image earlier:


1)Cell (Stores values over time intervals)
2)Input Gate
3)Output Gate
4)Forget Gate
Regulates how information enters and leaves cells

Forget gate: This gate concludes which information needs to be retained in memory and which information can be ignored. The current input information and hidden information is processed through a function (sigmoid), which generates a value that lies in the range of 0 to 1. This is used to decide whether the older output is required.
Input Gate: This gate performs the below operations that modify the status of cells: Firstly, current state and hidden state are processed by a sigmoid function that transforms it into a value that lies in the range of 0 for important and 1 for not important. Furthermore, this same information is processed by a tanh function which will create a vector that contains all the possible values in the range of -1 and 1. The output values can then be multiplied point-by-point.
Output Gate: The main responsibility of this gate is to tell the lstm what it would output. It combines the information of the updated memory state with the previous combined input and hidden state information to generate new information Sigmoid activation function is applied to the hidden and input state information. Tanh activation function is also applied on the updated memory state information. The outputs from both these processes are used for dot product multiplication and based upon the resulting value, it resolves the information needed to be carried by the hidden state which is then used for prediction.
Cell State: Here it gathers necessary information from input and forget gate and then, it stores this information in what is called memory state or cell state. A vector is calculated and its cross product is done with the previous cell state to determine if the values need to be dropped or not. The cell state can be thought of as a conveyor belt through which information flows unchanged.

C. *Steps and Equations*



As a first step, we need to figure out what necessary information needs to be retained in memory and what needs to be removed. This choice is produced by the following sigmoid layer (**forget gate**).

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

Next step, the **input gate** identifies which values require updating, after which a tanh layer generates a vector of values(C_t). These two are then combined to do an update to the state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{1}$$
$$\check{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{2}$$

The previous cell state (C_{t-1}), is updated to the new C_t state.

For this, we take the product of the past state with f_t, thereby forgetting the earlier-decided-to-be-forgotten things. To this we add $\check{C}_t$ * i_t.

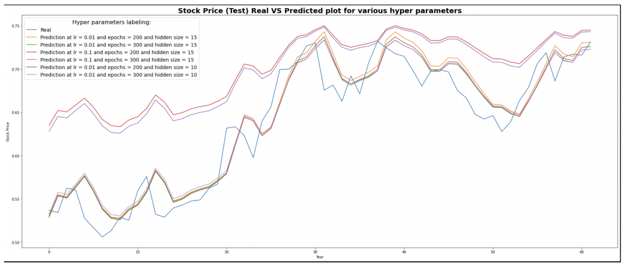
$$C_t = f_t * C_{t-1} + i_t * \check{C}_t \tag{1}$$

In last step, finally, we determine what needs to be output which will depend upon the cell state (it will be a modified version of the cell state). For this we apply the sigmoid activation function on the cell state and then subsequently apply the tanh activation (restricting the values between -1 and 1) and do a cross product with the result of the sigmoid layer.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{1}$$

$$h_t = o_t * tanh(C_t) \tag{2}$$

IV. Results and Conclusion



Experiment Number	Parameters Chosen	Results

1	Learning Rate = 0.01
	Hidden Layer Neurons = 15
	Number of Epochs = 200
		RMSE = 0.0256
		MAE = 0.0329

2	Learning Rate = 0.01
	Hidden Layer Neurons = 15
	Number of Epochs = 300
		RMSE = 0.0246
		MAE = 0.0318

3	Learning Rate = 0.1
	Hidden Layer Neurons = 15
	Number of Epochs = 200
		RMSE = 0.0758
		MAE = 0.1111

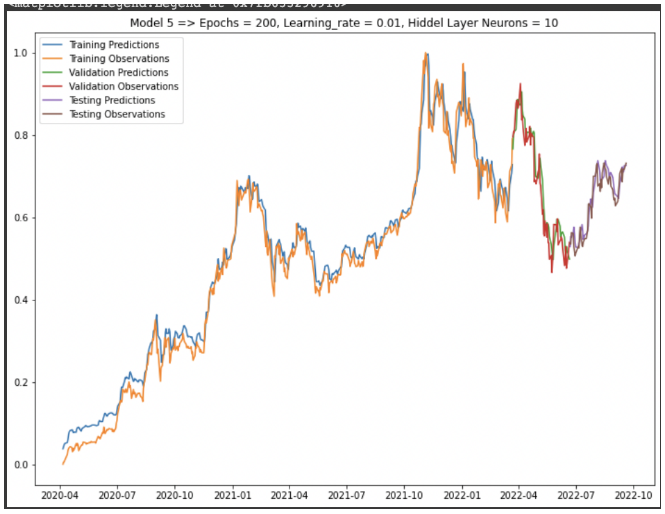
4	Learning Rate = 0.1
	Hidden Layer Neurons = 15
	Number of Epochs = 300
		RMSE = 0.0713
		MAE = 0.1041

5	Learning Rate = 0.01
	Hidden Layer Neurons = 10
	Number of Epochs = 200
		RMSE = 0.0244
		MAE = O.0315

6	Learning Rate = 0.01
	Hidden Layer Neurons = 10
	Number of Epochs = 300
		RMSE = 0.0246
		MAE = 0.0323

From tabular results, we can see that we obtained best accuracy with learning rate = 0.01, having hidden layer size of 10 neurons and training the model for 200 epochs. From this we can observe and conclude that a smaller learning rate helps train the model better and having less hidden layer neurons makes it more stable. Because having more hidden layers makes the model more complex and thus it is prone to overfitting.

From the tabular data, it can also be observed that Model 5 gives the best results with the lowest RMSE value.



References

[1] **Introduction to LSTM units in RNN**
https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

[2] **Understanding LSTM Networks**
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] **What Is Momentum? Definition in Trading, Tools, and Risks**
https://www.investopedia.com/terms/m/momentum.asp

[4] **Trading Volume: What it is and how it affects stocks**
https://seekingalpha.com/article/4437223-what-does-volume-mean-stocks

[5] **Essentials of Deep Learning : Introduction to Long Short Term Memory**
https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

[6] **LSTM Neural Network from Scratch**
https://www.kaggle.com/code/navjindervirdee/lstm-neural-network-from-scratch

[7] **What is time series forecasting?**
https://machinelearningmastery.com/time-series-forecasting/

[8] **How to convert a time series to a supervised learning problem**
https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

[9] **Recurrent Neural Networks**
https://www.ibm.com/cloud/learn/recurrent-neural-networks

[10] **Long term Short memory**
https://en.wikipedia.org/wiki/Long_short-term_memory