Assignment 3: Recurrent Neural Networks for Stock Price Prediction

Yuan Tian
a1818751@student.adelaide.edu.au

1. Introduction

In dealing with the complex nature of the stock market, a variety of methods and tools have been adopted to make predictions on the stock market. Having a high accuracy in predicting future stock prices is essential, as it may help investors to gain a higher yield of profit through stock investments by choosing the stocks that have the potential for a higher return based on the predictions. Machine learning is one of the methods used for stock return predictions in addressing the uncertainty of stock prices. Although different machine learning techniques have been utilised for predicting stock market over the years, the deep learning models are commonly used these years as a result of the increased amount of data and expectations in having higher accuracy in predictions. Evidence has shown that the deep learning models are more advantageous in its accuracy and prediction speed, compared to traditional machine learning methods[1]. The aim of this paper is to understand RNN and implement it to improve accuracy of predicting stock price. LSTM has been chosen to perform the task and the result of several tests has proved that LSTM has achieved great accuracy of predicting stock price.

2. Background

What makes RNNs to be called recurrent is that the same task is performed by RNNs for every sample based on the results from the past computations. RNNs are also considered to have a "memory" which transfers information between time steps. Theoretically speaking, RNNs can deal with long samples of a number of time steps. This has made RNNs particularly appropriate for stock prices in which information is transferred between price points. This implies that the price information from long time ago or the same day in the previous year still involve its remaining information to the current price. While it cannot be neglect that long samples make computations more time consuming and might be unnecessary [2].

Long-Short-Term Memory (LSTM) is one of the Recurrent Neural Networks using deep learning algorithms. The recurrent nature of the LSTM can be seen in its architecture with feedback connections. The main advantage of LSTM is that it has the capability to process the whole sequence of data. The architecture of LSTM is made up of four main components, including the cell, input gate, output gate and forget gate. The cell serves for remembering values across arbitrary time intervals, and the three gates serves for controlling the flow of information coming in and out of the cell. The responsibility of the cell of the model is to record the dependencies between the elements in the input sequence. The input gate is in charge of controlling how much new data comes into the cell, the forget gate is responsible for controlling how much value can remain in the cell, and the output gate serves for controlling how much value from the cell is utilised to complete the LSTM unit's output activation [3].

Yet some variants of the LSTM model whose architecture does not include the output gate can be found, e.g., Gated Recurrent Units (CRUs). LSTM Networks are mostly utilitsed on time-series data for the purposes of classifying, processing, and predicting. Why it is popular in time-series application is because multiple lags of uncertain duration between significant events in a time series can exist [4].

3. A Description of Method

LSTM has been chosen to perform the task which is to predict stock price based on Google stock price dataset from Kaggke. Therefore, it is necessary to explain the algorithms involved in detail.

Firstly, LSTM requires a 3-D array as the inputs which are tensor, time step and feature. The 1-D array can be converted to 3-D array by using np.reshape(samples, time steps, features). In theory, a long sample with many time steps can be handled in RNNs. However, it is a very time consuming process for a long sample and it is not necessary. For example, if there are 5 inputs (*xt-4, xt-3, xt-2, xt-1, xt*) and 2 outputs *yt+1* and *yt+2* in X_train data, then 5 hidden layers with 5 input time steps are consisted in the network.

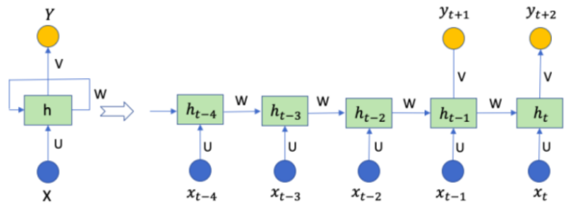


Figure 1 LSTM Unroll Network

The figure above is the process of RNN unrolls to a fully connected neural network. From the figure, Xt-4 to Xt represents the time steps of the sample and this is the vector of number of features. One feature will result as the dimension is 1 for Xt-4 to Xt. The memory of the network is named unit which is ht-4 to ht in the figures and it is the hidden state of time step.

$$h_t = f(U \cdot x_t + W \cdot h_{t-1} + b_t)$$

When iterates through the process previous hidden state to next hidden state, RNN encodes some information and this will help to predict from the current sequence. From the formula above, f(.) is the activation function which is tanh or Rectified Linear Unit. This is because the tanh or ReLU function can prevent infinity result by transform output between 0 and 1. Yt+1 and yt+2 are the outputs and bt is the noise term. Normally, the outputs are generated by the matrix product of activation function tanh and hidden neurons.

From the LSTM algorithm, units is defined as the dimensionality of the output space or the dimension of hidden layer. It is necessary to mention the number of neurons and also the capability of RNN which is determined by the hidden dimensionality. If the dimension of hidden layer is 32 and feature dimension is 1, therefore the dot product of U and xt has dimension 32*1.

In general, the control flow of LSTM network follows as a current input get concatenated with previous hidden state and then forget layer removes non relevant data. Then input layer decides which data values should add to new cell from the candidate layer. Therefore, vectors and previous state can be used to calculate all layers and then output is computed.

4. Experimental Analysis

The experiment was conducted to increase the accuracy of predicting stock price by using Google stock price data set from Kaggle. The experiment involve running several tests to show the final accuracy was successfully improved by implementing different LSTM architectures.

Firstly, the dataset need to be scaled and reshaped to 3-D array to use. The first LSTM model is built with sequential for neural network initialization and then followed by two long short-term memory layers. The last dense layer is also required as output layer and unit is 1 as the number of neurons which to give correct dimensionality to the target. The loss argument uses Root Mean Square Error for continuous variables.

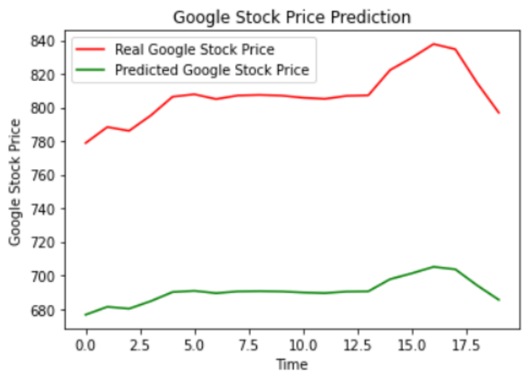


Figure 2: 2layers-Unit5 (Testing data)

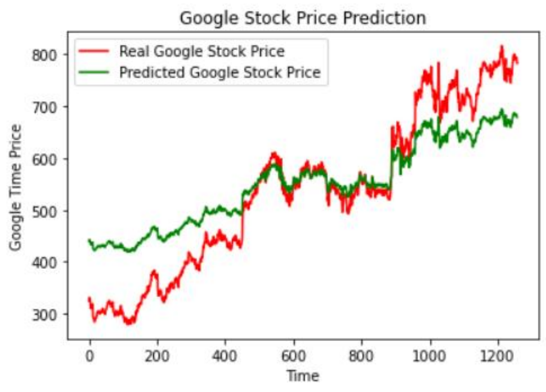
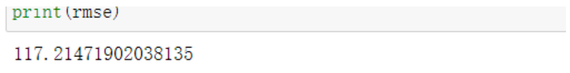


Figure 3: 2layers-Unit5 (Training data)



```
print(rmse)

117.21471902038135
```

The result of the basic LSTM model after 10 epochs has the running loss is between 0.0242 to 0.2908 and the root mean square error is 117.2 which result as 19.5% error. The above prediction graphs can better show the difference between real stock prices and predicted stock prices on 1 month's data and 4 years data. It can be obviously seen that predictions have similar trend to the real stock although the big caps are noticeable.

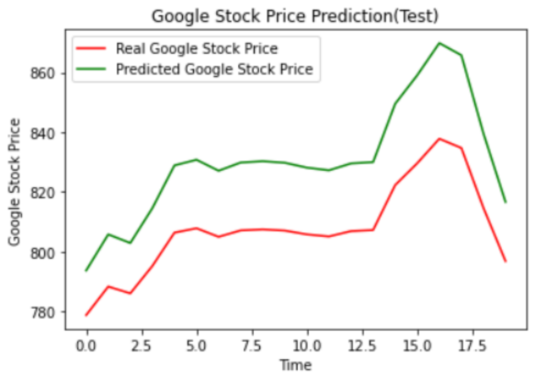


Figure 4: increase Units to 32 (Testing data)

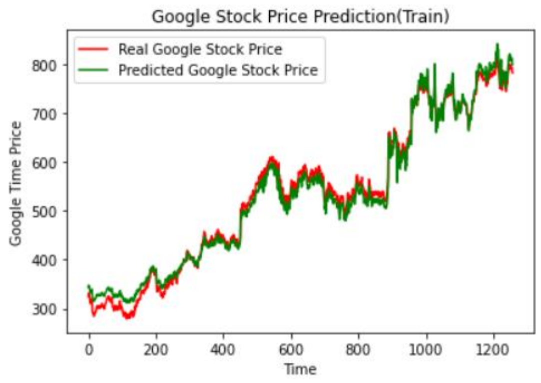
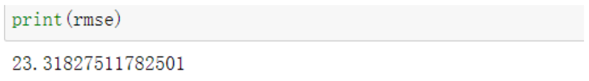


Figure 5: increase Units to 32 (Training data)



```
print(rmse)

23.31827511782501
```

Bonus:

The second test uses the same two LSTM layers but with increased units number 32 and then followed by the last dense layer. After 10 epochs, the root mean squared error of second test has decreased from 117 to 23 which is about 3.8%. The figure 2 test data graph has shown the predicted price line is much closer to the real price line compared to the first test and the training data graph shows two lines are almost matched. Therefore, the increase amount of units represents the width of the function to capture which can highly improve the accuracy of predictions.

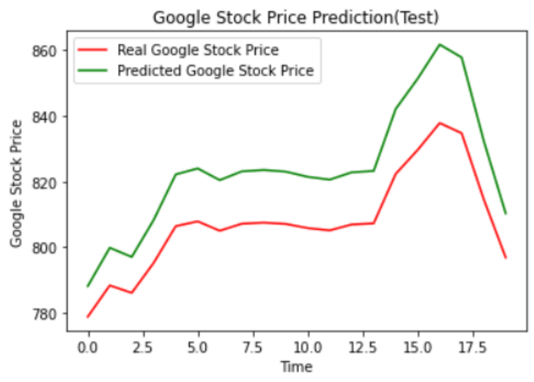


Figure 6: add extra dropout layers (Testing data)

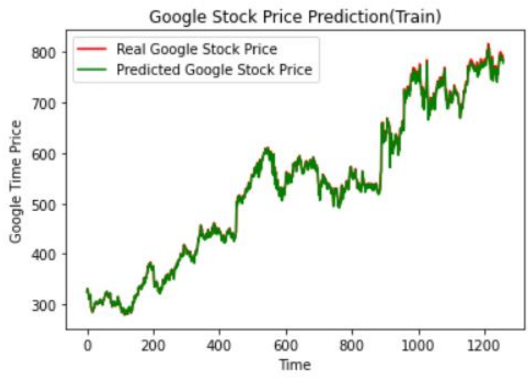


Figure 7: add extra dropout layers (Training data)

```
print(rmse)
```

```
16.543792718324507
```

The third test has modified with an extra dropout layer after each long short-term memory layer. It can be seen from the result that the RMSE has dropped from 23 to 16 which has small improvement to the accuracy. The dropout layer is to prevent overfitting, it can be assumed that dropout layer can have significant change for much more complex dataset.

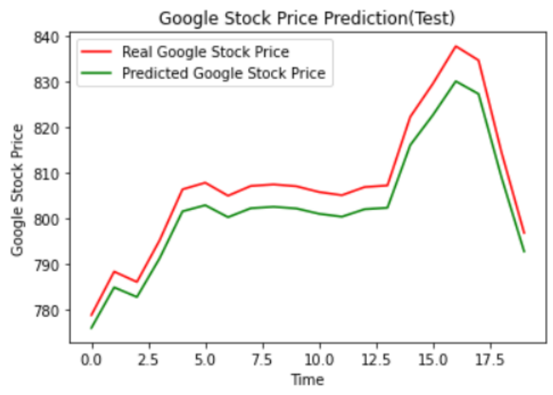


Figure 8: increase epochs (Testing data)

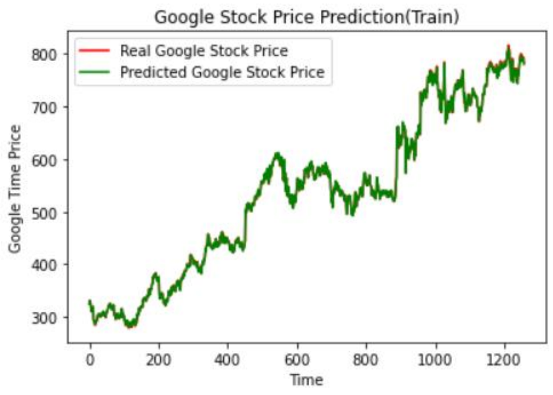


Figure 9: increase epochs (Training data)

```
print(rmse)
```

```
5.131120665768993
```

The fourth test has increased epochs from 10 to 100 and RMSE has dropped from 16 to 5. The result indicates increased epochs will have significant improvement on predictions' accuracy. The prediction graphs above has demonstrated that the current LSTM had great performance on predicting stock price.

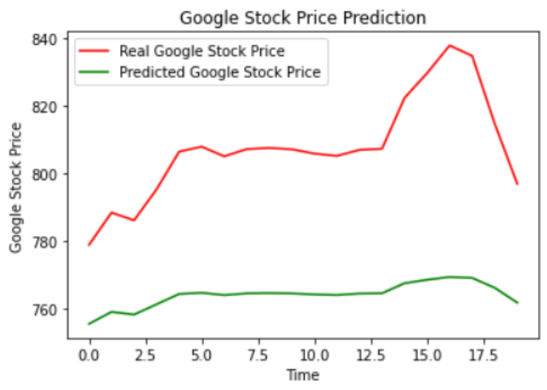


Figure 10: add 3 more LSTM layers (Testing data)

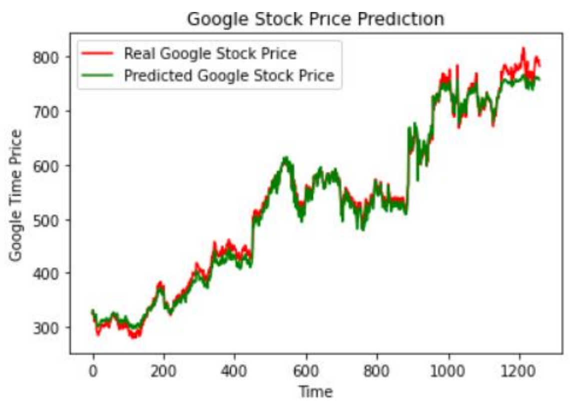


Figure 11: add 3 more LSTM layers (Training data)

```
print(rmse)
```

```
45.03135527028927
```

The fifth test was conducted to find the influence of adding 5 LSTM layers and each LSTM layer has one dropout layer. After 100 epochs, the result has shown the Root Mean Squared Error has increased from 5 to 45. However, it cannot be concluded that increased layers will definitely decrease the predictions accuracy because this dataset is not complex enough. More LSTM layers will increase the depth of neural network which can increase the complexity and cause overfitting.

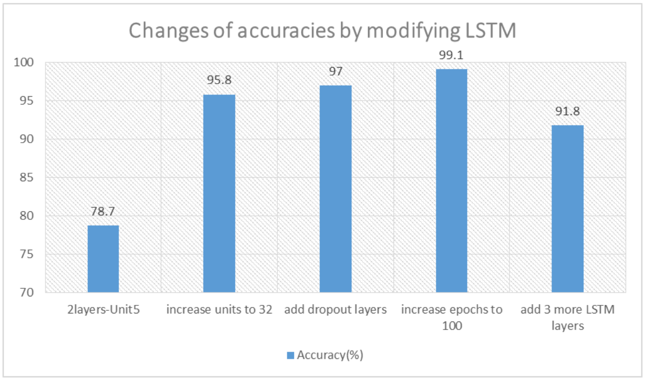


Figure 12: Changes of accuracies by modifying LSTM

The result of five tests has clearly addressed the changes of accuracies by modifying LSTM structure. Increasing units, adding dropout layers and increasing epochs have increased accuracy in various degrees, however increasing depth of network resulted some drawbacks on accuracy. The best tested LSTM model consisted 2 layers of hidden layers with 32 units and followed by dropout layers, then fit the model for 100 times epochs. Therefore, the highest accuracy can be achieved as 99.1%.

5. Code

https://github.com/Yuan-Tian2020/RNN-LSTM-predictGoogle.git

git@github.com:Yuan-Tian2020/RNN-LSTM-predictGoogle.git

6. Conclusion

This paper has discussed the application of RNN models and specifically focused on LSTM to perform the task which is to predict the Google stock price. The idea of conducting this experiment is to understand the functions of each parameters in algorithms and gain knowledge to develop the LSTM model. The result of five tests have clearly indicated the importance of units, dropout layers, epochs and depth of hidden layers. The minimum Root Mean Square Error has achieved as 5.13 compare to the average stock price of 550. LSTM has shown the great performance on predicting continues variables such as stock price. Despite the experiment addressed more units, add dropout layers and more epochs can increase the accuracy, adding more LSTM layers still need to be discussed. It can be assumed that reduce the number of units in each layer when adding more hidden layers can effectively avoid overfitting problems. On the other hand, higher accuracy can still possibly achieved with right combination of depth (layers) and width (units).

References

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

[2] A. M. Rather, A. Agarwal, & V. N. Sastry, “Recurrent neural network and a hybrid model for prediction of stock returns”, *Expert Systems with Applications*, vol. 42, no.6, pp. 3234-3241, 2015.

[3] K. A. Althelaya, E. M. El-Alfy and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," *2018 9th International Conference on Information and Communication Systems (ICICS)*, pp. 151-156, 2018, doi: 10.1109/IACS.2018.8355458.

[4] M. O. Rahman, M. S. Hossain, T. S. Junaid, M. S. A. Forhad, &, M. K. Hossen, “Predicting prices of stock market using gated recurrent units (GRUs) neural networks,” *Int. J. Comput. Sci. Netw. Secur*, vol.19, no.1, pp. 213-222, 2019.