Stock Price Forecasting with Recurrent Neural Networks (LSTM)

Peng Gao
School of Natural Sciences and Mathematics
The University of Texas at Dallas
Richardson, Texas, USA

Yifei Li
School of Natural Sciences and Mathematics
The University of Texas at Dallas
Richardson, Texas, USA

Ying Wang
School of Natural Sciences and Mathematics
The University of Texas at Dallas
Richardson, Texas, USA

Yuming Ren
School of Natural Sciences and Mathematics
The University of Texas at Dallas
Richardson, Texas, USA

***Abstract*---Long short-term memory (LSTM) was invented in 1997 as a special type of Recurrent Neural Networks (RNN). Compared with the traditional RNNs, LSTM can effectively resolve the vanishing gradient issue and hence fit better with time series data prediction. Nowadays, LSTM is widely used in different deep learning and artificial intelligence areas, such as language and handwriting recognition, automatic caption, etc. In this report, we will review the mechanism of LSTM and try to implement the LSTM model to predict the stock price of Apple Inc. Furthermore, we will discuss how to interpret the effectiveness of stock market forecasting with LSTM.**

I. INTRODUCTION

For thousands of years, people try to use their previous experiences to explain the events that are currently happening. Moreover, people wish to use their experiences to predict future events. If we look at these experiences as data, we can easily find the similarity of the predicting work between some machine learning jobs now and what people did in the past thousand years.

A. Time series data

To predict future events, one type of the most useful information is time series data. Time series data is the data observed or collected in different time points. Time interval between two immediately sequential points could be hour, day, year or any discrete scale in a dataset. Since it is ordered with time sequence, some statistical properties will vary over time, such as mean, range and standard deviation. Good examples of time series data are stock price, weather temperature, disease infected population, etc.

Based on historical data, time-series forecasting predicts future values. It is widely applied in financial, medical and many other different research domains.

B. RNN and LSTM

When using machine learning methods to solve time-series forecasting problems, the common neural network won't work well since it cannot retain information along time sequences. People find the Recurrent Neural Networks (RNN) based on David Rumelhart's research in 1986. [1] RNN is a type of artificial neural networks whose hidden state can pass information along time steps and hence retain the sequence information.

However, a main issue of traditional RNNs is that the gradient can vanish during RNN using back propagation. In 1997, the long short-term memory (LSTM) was invented by Sepp Hochreiter and Jürgen Schmidhuber.[2] LSTM is a special RNN architecture that can solve the "vanishing gradient" issue by regulating information flow.

II. MECHANISM OF LSTM RNNS

Traditional RNN usually have very simple structures such as a single activation function in each module. (Fig. 1) Easily to see there is not much regulation of what kind or how much of information will be passed from one unit to the next one.

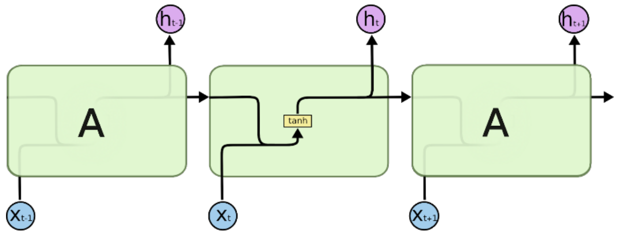


Fig. 1. Traditional RNN with single tanh layer[3]

A. Forward Pass

LSTMs have more complex structures. A typical LSTM cell is composed of five essential parts, cell state, hidden state and three types of gate: **forget gate, input gate** and **output gate**. These gates are actually regulators which control the flow of information inside LSTMs.(Fig. 2).

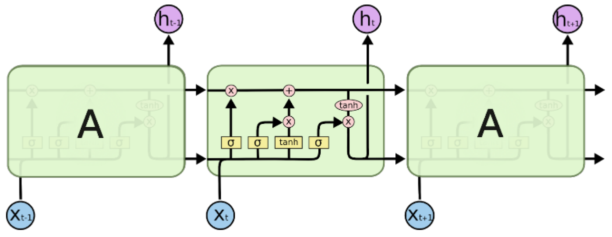
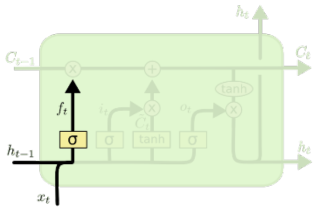


Fig. 2. LSTM contains multiple activation layers[3]

In this diagram, the top horizontal line represents the cell state. Cell state can be regarded as a railroad, on which there are many stations (cells). When our information train travels along the railroad, at each cell some passengers step out the train and some new passengers join the journey.

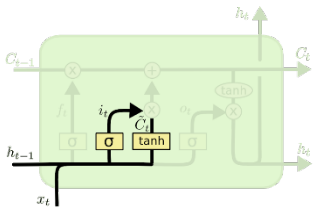
First part in LSTM is the "forget gate" if we break down the cell and analyze how it works step by step. When some input *Xt* and previous cell's output *ht-1*come into the cell, the forget gate with a sigmoid function will output a number between 0 and 1 to the cell state *Ct-1*. A 0 means "completely forget this information" and a 1 means "keep all the information".



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

Fig. 3. The Forget Gate[3]

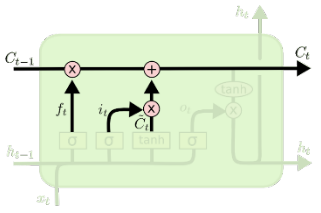
Second step is to decide how much new information will be added into the cell state. Another sigmoid function here will calculate the portion of new information will be added to the cell state. Next, a tanh function will output a vector which will be added in cell state in the later step. In our "train-passengers" interpretation, this step can be seen as some new passengers arrive at the station but only a part of them have valid tickets.



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

Fig. 4. The Input Gate[3]

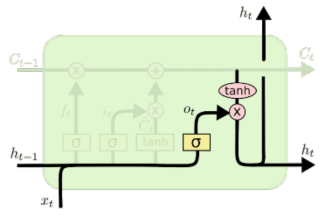
The third step is to combine the results from the previous two steps and update the cell state, Ct-1, to Ct. How much information to be dropped was decided by the "forget gate" while how much new information to be added was decided by the "input gate". Now the information train is ready to take off and heading to the next station.



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

Fig. 5. Updating Cell State with New Information[3]

The final step in LSTM is to use the "output gate" to filter out unwanted information. Similar to the "input gate", the "output gate" includes a sigmoid function and a tanh function. The sigmoid function will determine what information of the cell state should be output, while the tanh function will rescale the cell state to be between -1 to 1. Multiplying the results gives the output, ht. Notice in this step, the cell state does not change.



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

Fig. 6. The Output Gate[3]

B. Backpropagation

In this LSTM model, we need to update two parameters including weights and bias (offsets) which was conducted in the backward propagation of errors, i.e. backpropagation. All nodes of LSTM were defined same as the description in forward pass part.

The aim of backpropagation is mainly to update the parameters W and b, and then to decrease the total loss:

$$L = \sum_{t=1}^{T} ||h_t - y_t||^2$$

Therefore, we need to do the partial derivative to the loss function with respect to parameters W and b of each components of LSTM model, in other words, we should calculate $\frac{\partial L}{\partial W_f}$ and $\frac{\partial L}{\partial b_f}$ for the forget gate $f(t)$, $\frac{\partial L}{\partial W_i}$ and $\frac{\partial L}{\partial b_i}$ for the input gate $i(t)$, $\frac{\partial L}{\partial W_o}$ and $\frac{\partial L}{\partial b_o}$ for the output gate $o(t)$, $\frac{\partial L}{\partial W_g}$ and $\frac{\partial L}{\partial b_g}$ for the input modulation gate $g(t)$, $\frac{\partial L}{\partial W_c}$ and $\frac{\partial L}{\partial b_c}$ for the current cell state $c(t)$, $\frac{\partial L}{\partial W_h}$ and $\frac{\partial L}{\partial b_h}$ for the current hidden state $h(t)$. We should calculate the partial derivative in taking full advantage of two equations:

$$c(t) = g(t) \odot i(t) + c(t-1) \odot f(t)$$
$$h(t) = c(t) \odot o(t)$$

"$\odot$" refers to the component-wise multiplication for matrices and the concatenating of input vectors $x(t)$ and previous hidden state value $h(t-1)$ is denoted as $[x(t), h(t-1)]$ in all formulas.

The mathematics of each partial derivative is shown as follows:

1) *For the forget gate:*
$$f(t) = sigmoid\{W_f[x(t), h(t-1)] + b_f\}$$
2) *For the input gate:*
$$i(t) = sigmoid\{W_i[x(t), h(t-1)] + b_i\}$$
3) *For the output gate:*
$$o(t) = sigmoid\{W_o[x(t), h(t-1)] + b_o\}$$
4) *For the input modulation gate:*
$$g(t) = tanh\{W_g[x(t), h(t-1)] + b_g\}$$
5) *The partial derivative w.r.t* W_f *is:*
$$\frac{\partial L(t)}{\partial W_f} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial W_f}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial \left\{sigmoid\{W_f[x(t), h(t-1)] + b_f\}\right\}}{\partial W_f}$$
$$= \frac{\partial L(t)}{\partial c(t)} sigmoid'\{W_f[x(t), h(t-1)] + b_f\} \cdot [x(t), h(t-1)]$$

6) *The partial derivative w.r.t b_f is:*

$$\frac{\partial L(t)}{\partial b_f} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial b_f}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [g(t) \odot i(t) + c(t-1) \odot f(t)]}{\partial b_f}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [sigmoid\{W_f[x(t), h(t-1)] + b_f\}]}{\partial b_f} c(t-1)$$

7) *The partial derivative w.r.t W_i is:*

$$\frac{\partial L(t)}{\partial W_i} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial W_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [g(t) \odot i(t) + c(t-1) \odot f(t)]}{\partial W_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [sigmoid\{W_i[x(t), h(t-1)] + b_i\}]}{\partial W_i} g(t)$$

8) *The partial derivative w.r.t b_i is:*

$$\frac{\partial L(t)}{\partial b_i} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial b_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [g(t) \odot i(t) + c(t-1) \odot f(t)]}{\partial b_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [sigmoid\{W_i[x(t), h(t-1)] + b_i\}]}{\partial b_i} g(t)$$

9) *The partial derivative w.r.t W_o is:*

$$\frac{\partial L(t)}{\partial W_o} = \frac{\partial L(t)}{\partial h(t)} \frac{\partial h(t)}{\partial W_o} = \frac{\partial L(t)}{\partial h(t)} \frac{\partial [c(t) \odot o(t)]}{\partial W_o}$$
$$= \frac{\partial L(t)}{\partial h(t)} \frac{\partial [o(t)]}{\partial W_o} c(t)$$
$$= \frac{\partial L(t)}{\partial h(t)} \frac{\partial [sigmoid\{W_o[x(t), h(t-1)] + b_o\}]}{\partial W_o} c(t)$$

10) *The partial derivative w.r.t b_o is:*

$$\frac{\partial L(t)}{\partial b_o} = \frac{\partial L(t)}{\partial h(t)} \frac{\partial h(t)}{\partial b_o} = \frac{\partial L(t)}{\partial h(t)} \frac{\partial [c(t) \odot o(t)]}{\partial b_o}$$
$$= \frac{\partial L(t)}{\partial h(t)} \frac{\partial [o(t)]}{\partial b_o} c(t)$$
$$= \frac{\partial L(t)}{\partial h(t)} \frac{\partial [sigmoid\{W_o[x(t), h(t-1)] + b_o\}]}{\partial b_o} c(t)$$

11) *The partial derivative w.r.t W_a is:*

$$\frac{\partial L(t)}{\partial W_g} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial W_g}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial \{g(t) \odot i(t) + c(t-1) \odot f(t)\}}{\partial W_g}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial \{tanh\{W_g[x(t), h(t-1)] + b_g\}\}}{\partial W_g} i(t)$$
$$= \frac{\partial L(t)}{\partial c(t)} tanh'\{W_g[x(t), h(t-1)] + b_g\} \cdot [x(t), h(t-1)] \cdot i(t)$$

12) *The partial derivative w.r.t b_a is:*

$$\frac{\partial L(t)}{\partial b_g} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial b_g}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial \{g(t) \odot i(t) + c(t-1) \odot f(t)\}}{\partial b_g}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial \{\tanh\{W_g[x(t), h(t-1)] + b_g\}\}}{\partial b_g}$$

13) *For the current cell state*

$$c(t) = g(t) \odot i(t) + c(t-1) \odot f(t):$$
$$\frac{\partial L(t)}{\partial W_i} = \frac{\partial L(t)}{\partial c(t)} \frac{\partial c(t)}{\partial W_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [g(t) \odot i(t) + c(t-1) \odot f(t)]}{\partial W_i}$$
$$= \frac{\partial L(t)}{\partial c(t)} \frac{\partial [sigmoid\{W_i[x(t), h(t-1)] + b_i\}]}{\partial W_i} g(t)$$

14) *For the current hidden state $h(t) = c(t) \odot o(t)$:*

$$\frac{\partial L(t)}{\partial h(t)} = \frac{\partial [\sum_{t=1}^{T} ||h_t - y_t||^2]}{\partial h(t)}$$

III. STOCK PRICE FORECASTING MODEL

A. *Data Description*

To demonstrate one of the applications of LSTM, we constructed a machine learning model to predict the stock price of Apple Inc.. The dataset we used was from Yahoo Finance (https://finance.yahoo.com/). It contains 5032 instances which cover all trading days from 11-24-2000 to 11-19-2020. In each instance, five variables are included: Close, Volume, Open, High, Low. Fig. 7 shows the actual closing price of the complete data set.

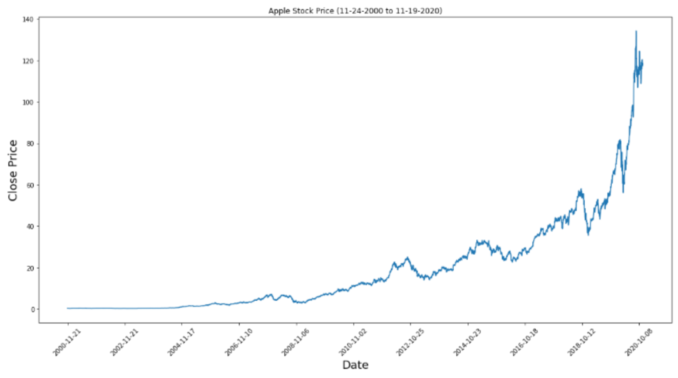


Fig. 7: The Close Price of Dataset

B. *Method*

In the data preprocessing stage, the dataset was split into training and the testing parts. The training part of the dataset we used is from date 2000-11-21 to 2017-12-29, and the testing part of the dataset is from 2018-01-01 to 2020-11-19.

We chose to use all five variables of a trading day as the predictor variables and the predicted variable would be the closing price. There are four different parameters in this model, namely learning rate (alpha), epoch, time steps and dimension.

Next, the training data was pushed through a single layer LSTM and hence the model was trained and ready to use as a complete and applicable model. Then, the testing data was input into the model to verify the accuracy of the model. The learning rate, epoch number, timesteps and dimensions were tuned with different combinations to optimize our model.

IV. RESULTS AND ANALYSIS

A. *Effect of Learning Rate (Alpha)*

The model performs better at a lower learning rate as we can see from the graph as well as RMSE when alpha drops from 0.1 to 0.01. (Fig. 8 and Fig. 9) However, the difference is not that dramatic when alpha drops from 0.01 to 0.001. We decided to use alpha=0.01 for later procedure. (Fig. 9 and Fig.10)

1) Alpha=0.1, time steps=10, n_dim=100, epoch=5.

RMSE: 60.653830524805024.

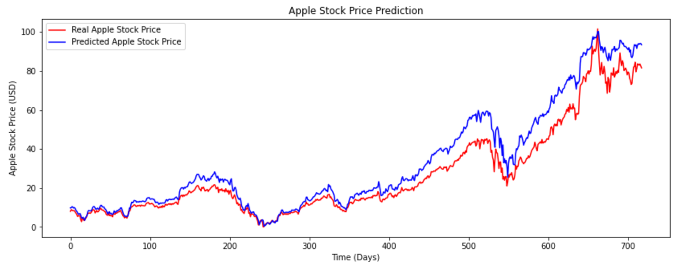


Fig. 8: Comparison of Actual and Predicted Price (Learning Rate = 0.1)

2) Alpha=0.01, timesteps=10, n_dim=100, epoch=5.

RMSE: 19.975323620689323.

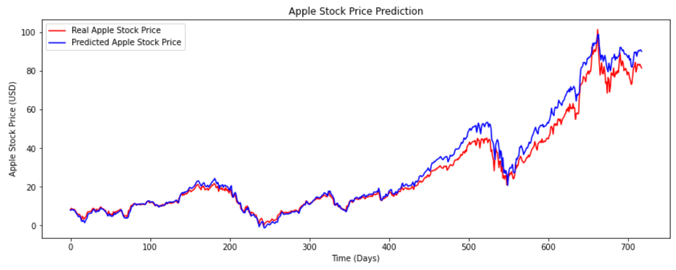


Fig. 9: Comparison of Actual and Predicted Price (Learning Rate = 0.01)

3) Alpha=0.001, timesteps=10, n_dim=100, epoch=5.

RMSE: 10.120671269665076.

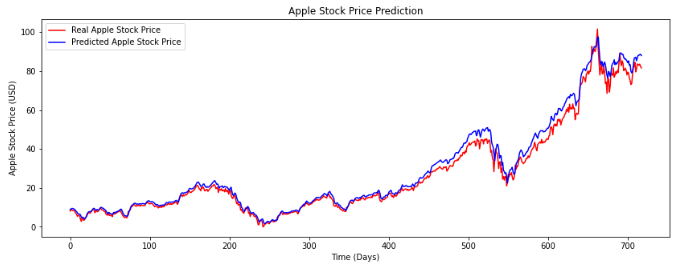


Fig. 10: Comparison of Actual and Predicted Price (Learning Rate = 0.001)

B. *Effect of Epoch Number*

One epoch means all instances go through the model for one time. We tested different epoch numbers at 2, 5, 10 and 20. Results show the model performs better when the epoch number goes up as we can see from the graph as well as RMSE.

1) Alpha=0.01, timesteps=10, n_dim=100, epoch=2.

RMSE: 1474.2726270118721.

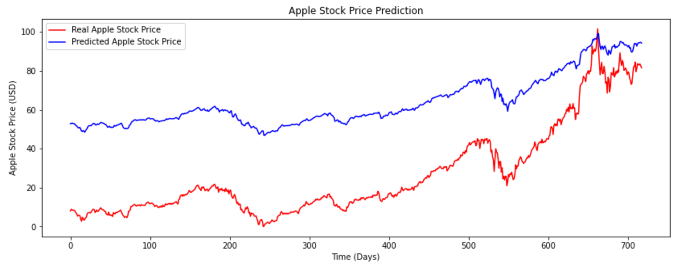


Fig. 11: Comparison of Actual and Predicted Price (Epoch = 2)

2) Alpha=0.01, timesteps=10, n_dim=100, epoch=5.

RMSE: 19.975323620689323.

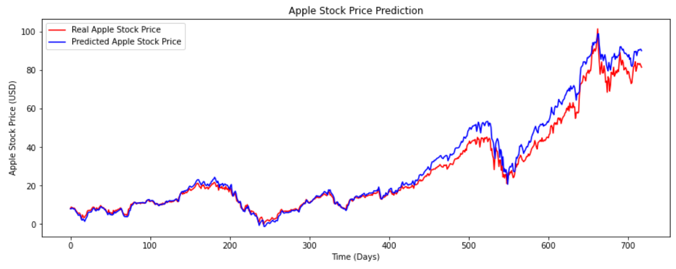


Fig. 12: Comparison of Actual and Predicted Price (Epoch = 5)

3) Alpha=0.01, timesteps=10, n_dim=100, epoch=10.

RMSE: 15.221252104112722.

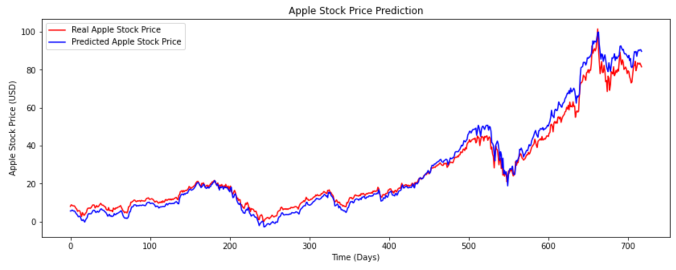


Fig. 13: Comparison of Actual and Predicted Price (Epoch = 10)

4) Alpha=0.01, timesteps=10, n_dim=100, epoch=20.

RMSE: 13.77123895638986.

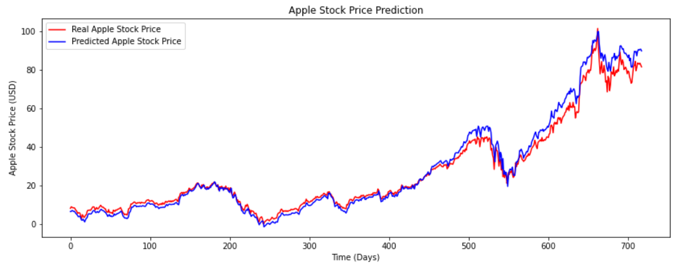


Fig. 14: Comparison of Actual and Predicted Price (Epoch = 20)

C. *Effect of size of timesteps*

Timesteps is how many days are used to predict the future price. We can see from the graphs and RMSEs that the bigger timesteps the better prediction and smaller RMSE. In another word, when we use more days to train the model, we have better prediction.

1) Alpha=0.01 timesteps=20, n_dim=100, epoch=5.
RMSE: 19.377145771248493

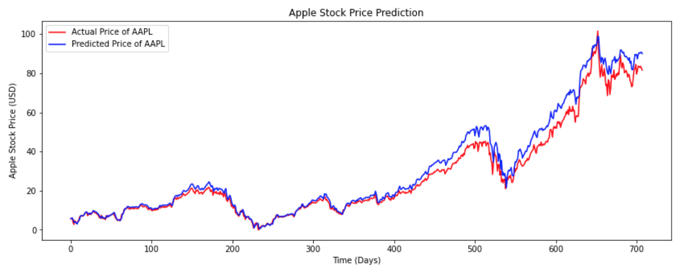


Fig. 15: Comparison of Actual and Predicted Price (Timesteps = 20)

2) Alpha=0.01 timesteps=50, n_dim=100, epoch=5
RMSE: 12.178593314964228

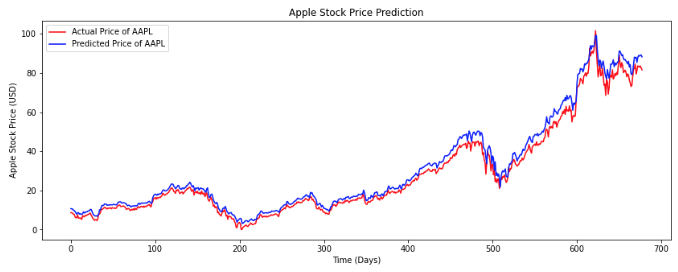


Fig. 16: Comparison of Actual and Predicted Price (Timesteps = 50)

3) Alpha=0.01 timesteps=100, n_dim=100, epoch=5
RMSE: 5.399319633533358

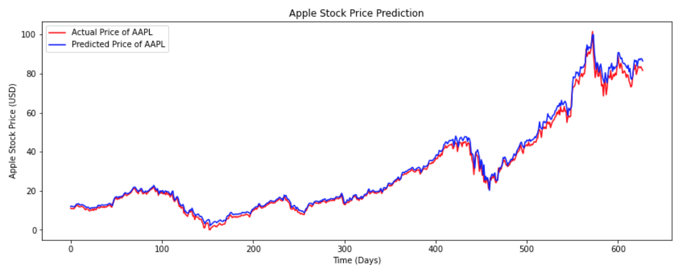


Fig. 17: Comparison of Actual and Predicted Price (Timesteps = 100)

D. Effect of size of dimensions

Our experiments show that changing the dimensions has no obvious effects to the RMSEs (see Table I).

TABLE I. SUMMARY OF RMSE OF DIFFERENT MODELS

Effect of hyperparameter	Dimension	Timesteps	alpha	epoch	RMSE
epoch	100	10	0.01	2	1474.27
	100	10	0.01	5	19.98
	100	10	0.01	10	15.22
	100	10	0.01	20	13.77
learning rate	100	10	0.1	5	60.65
	100	10	0.01	5	19.98
	100	10	0.001	5	10.12
timestep size	100	10	0.01	5	19.98
	100	20	0.01	5	19.38
	100	50	0.01	5	12.18
	100	100	0.01	5	5.4
dimension	100	10	0.01	5	19.98
	50	10	0.01	5	24.27
	10	10	0.01	5	22.44

After compiling the code, we can observe from the obtained data that when the values of all other parameters remain unchanged, the corresponding RMSE value becomes smaller as the learning rate becomes smaller. Also, with the increase of epoch, the value of RMSE gradually decreases. Therefore, of the parameters we ran, the model's performance was optimal when Alpha=0.01, dimensions=100, timesteps increases and epoch increases. The value of root mean squared derivation (RMSE) was decreasing follows this trend. Moreover, the two curves in the figures are relatively closer to each other. This means that the predicted stock price trend is very consistent with the real stock price.

V. CONCLUSION

Based on the model testing results, we can draw two conclusions. First, as a special RNN architecture, LSTM is capable of predicting stock price. Second, the testing results can be optimized at some extent by tuning learning rate, time steps and epoch numbers.

VI. FURTHER DISCUSSION

A. Is Stock Price Predictable?

Although the LSTM model we constructed in this project works well on forecasting the Apple stock price, investors should not make any investment decision solely based on the LSTM model prediction results. Besides the historical price, there are many other factors which have significant impacts on the stock market that should be considered, such as political election results, natural disaster, management personnel change and so on. Most of the time, occurrences of these events cannot be predicted in advance. Because there are many unpredictable factors when predicting stock price, some people hence believe that the past price movement or trend of a stock cannot be used to predict its future movement. This is well known as "Random Walk" theory, infers that the stock price movement is more like a random walk on the street. On the other hand, many mathematical/statistical models have been proved capable to predict future trends correctly. Thus, whether stock price is completely predictable is still a contradiction.

Again, the purpose of this project is to demonstrate how to apply LSTM to forecast future movement trends, not to give any investment suggestions.

B. Future Work

Other than stock price forecasting, LSTM is also well fit for any data embedded in time. In the past decade, LSTM outperformed traditional models and methods in speech recognition [4], handwriting recognition [5], and automatic image captioning [6], etc.

Further research can be done based on the stack LSTM model in this project. Theoretically, Adam and other optimizers will help the mean square error decrease more efficiently. More tests need to be done to decide which optimizer works best for a certain type of data. How to improve the forecasting performance by tuning the hyperparameters is another interesting topic which needs more investigation.

REFERENCES

[1] Williams, Ronald J.; Hinton, Geoffrey E.; Rumelhart, David E. (October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088): 533–536.

[2] Hochreiter, Sepp; Schmidhuber, Jürgen (1997-11-01). "Long Short-Term Memory". Neural Computation. 9 (8): 1735–1780.

[3] “Understanding LSTM Networks,” Understanding LSTM Networks -- colah's blog. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 22-Nov-2020].

[4] Fernández, Santiago; Graves, Alex; Schmidhuber, Jürgen (2007). An Application of Recurrent Neural Networks to Discriminative Keyword Spotting. Proceedings of the 17th International Conference on Artificial Neural Networks. ICANN'07. Berlin, Heidelberg: Springer-Verlag. pp. 220–229.

[5] Graves, Alex; Liwicki, Marcus; Fernandez, Santiago; Bertolami, Roman; Bunke, Horst; Schmidhuber, Jürgen (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 31 (5): 855–868.

[6] Vinyals, Oriol; Toshev, Alexander; Bengio, Samy; Erhan, Dumitru (2014-11-17). "Show and Tell: A Neural Image Caption Generator". arXiv:1411.4555.