Long Short-Term Memory for Time Series Forecast: A case study of Google Stock Price

Anh Duc Le – Cork – June 2020

Table of Contents

1. Introduction

Stock price prediction is one of the most recurrent tasks of any investors. A number of techniques dealing with stock price prediction include traditional statistical methods such as parametric autoregressive (AR) or autoregressive models with integrated moving average (ARIMA) and modern statistical ones such as machine learning or deep learning. Singling out one of the most well-acclaimed deep learning techniques, this report will focus on explaining how Long Short-term Memory (LSTM) is used for predicting Google Stock Price.

2. LSTM Research

2.1. Theoretical background

Stock price prediction is the time series forecast which is formularised as the prediction of a time series $\{y_1, y_2, \dots\}$ at the time i based on its previous data $y_{i-1}, y_{i-2}, \dots$. If there is a vector $\boldsymbol{x} = \{y_{i-k}, y_{i-k+1}, \dots, y_{i-1}\}$ where $i = \{k, \dots, n\}$, the goal is to find a function $f(\boldsymbol{x})$ so that $\widehat{y_i} = f(\boldsymbol{x})$ is as close to the ground truth y_i as possible (Gamboa, 2017).

Theoretically, LSTM is suitable for stock price prediction thanks to its ability of capturing historical dependencies by customising the hidden layer neuron as the memory block featuring recurrently connected modules called the memory cell and gates (Figure 1). The memory cell is responsible for remembering the temporal state of the neural network; meanwhile, the gates control the pattern of information flow. There are three main gates consisting of input gates, output gates and forget gates. Input gates decide how much new information is used in the memory cell. Forget gates controls how much information of the previous memory cell still remains in the current memory cell through recurrent connection. Output gates manipulate how much information is used to compute the memory cell's output activation which will flow into the rest of the neural network (Finsveen, 2018). The sophisticated coordination between the memory cell and the gates

grants LSMT an outstanding ability to predict time-series with long-term dependences (Hua et al., 2019). Mathematical details of how LSTM works is presented in Appendix 1.

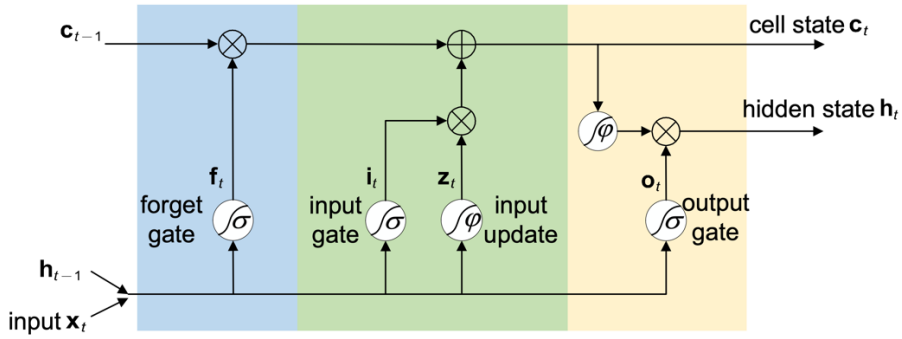


Figure 1: LSTM memory block

2.2. Work with LSTM

LSTM models are built with the help of Keras library. Several hyperparameters constructing a LSTM model are listed as follows:

- Number of hidden layers: there is no rule of thumb of how to choose number of layers. Usually, one hidden layer work with simple problems, two layers have shown to be enough to detect more complex features, more layers can be better but also harder to train.
- Number of input neurons for each layer: there is no specific rule of choosing the number of input neurons. Nevertheless, it is recommended as:

$$N_h = \frac{N_s}{(\alpha \times (N_i + N_o))}$$

Where: N_h is the number of hidden input neurons, N_i is the number of input neurons, N_o is the number of output neurons, N_s is the number of samples in training data & α is the arbitrary scaling factor from 2 to 10.

- Drop out layers: This layer will help to prevent overfitting by ignoring randomly selected neurons during training, and hence reduces the sensitivity to the specific weights of individual neurons. 20% is often used as a good compromise between retaining model accuracy and preventing overfitting.

- Number of Epochs: There is no specific rule for this. A few numbers of epochs lead to underfitting. Too many epochs result in overfitting.
- Batch size: There is no clear guide for this. Larger batch sizes result in faster progress in training, but don't always converge as fast. Smaller batch sizes train slower, but can converge faster. The common batch size is 32 but it is not always the case.
- Optimiser: Commonly used optimisers are "adam", i.e. Adaptive Moment Estimation provides both the smart learning rate annealing and momentum behaviours of the algorithms, and "nadam", i.e. being similar to "adam" but with Nesterov momentum instead of ordinary momentum.
- Loss function: For regression problem, the common loss functions are "mean squared error" and "mean absolute error".

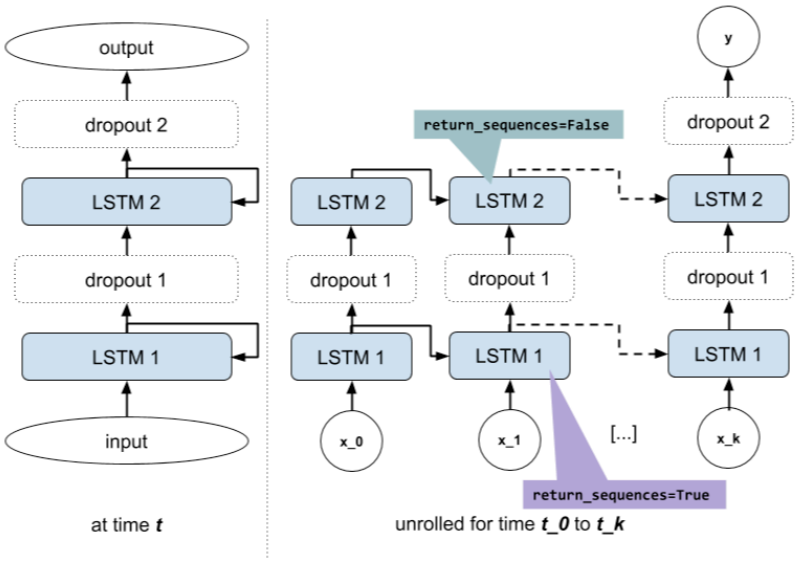


Figure 2. LSTM Architecture

3. Methodology

3.1. Data pre-processing

The dataset was first aggregated from Yahoo Finance from the period of 2004 to present. It includes several features such as opening price (open), closing price (close) highest highest price the stock traded at (High), how many stocks were traded (Volume) and closing price adjusted for stock splits and dividends (Adjusted Close) (Table 1). The

dataset is split into training set (before 31st Dec 2018) and testing set (after 31st Dec 2018) (Figure 3). Before training the models, the date column is deleted and the rest of the columns are rescaled between zero and one so that the models would not overestimate one feature to the others due to different scales. All of the remaining columns are used for building models from the training set. In the training set, each of the rolling window of 60 days is trained to predict the adjusted closing price in the very next day. The sample principle will be applied for the testing set when making predictions.

Date	Open	High	Low	Close	Adjusted Close	Volume
May 01, 2020	1,324.09	1,351.43	1,309.66	1,317.32	1,317.32	2,443,600
Apr 30, 2020	1,331.36	1,350.00	1,321.50	1,346.70	1,346.70	2,792,100
Apr 29, 2020	1,345.00	1,360.15	1,326.73	1,342.18	1,342.18	5,417,900
Apr 28, 2020	1,283.20	1,284.76	1,230.38	1,232.59	1,232.59	4,035,000
Apr 27, 2020	1,292.00	1,294.10	1,265.06	1,270.86	1,270.86	2,209,300

Table 1. Google Stock Price from 27 Apr to 1 May 2020

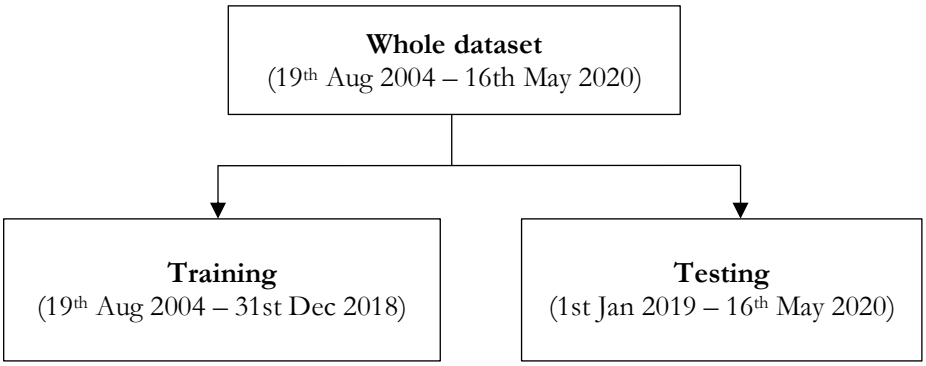


Figure 3. Training/Validation/Testing Splits

3.2. Building Model

The building model process took a lot of time to select the right hyperparameters that enable model to yield desirable results. Noticeably, number of hidden neurons, number of epochs and number of batches are the trickiest ones when customising the model. Firstly, Different number of hidden neurons, which were experimented, are from

10, 20, 50, 60, 80, 100, 200 to 500. The best numbers are 60 and 80. Secondly, different numbers of epochs and batches strongly manipulate the prediction results. In extreme scenarios, there is no correlation shown between the predicted stock price and the actual price if those numbers are below 10 and 10 or above 200 and 500 for epochs and batches respectively.

3.3. Evaluation Metrics

R-squared telling how variations of the predicted values is explained by the features and Mean square error measuring the average of the squares of the errors are used in the report. R-squared ranges from 0 (worst) to 1 (best) and Mean square error is from 0 (best) to infinity (worst). The formulas of the evaluation metrics are below:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

4. Evaluations and Conclusions

Figure 4 shows the final LSTM model. Figure 5 shows the stock price predictions during the period between 1st Jan 2019 to 16th May 2020. The results are promisingly desirable with the R-squared of 0.8618 meaning 86.18% of the predictions' variations explained by the features. Meanwhile, mean square error is quite small with 0.000029. To a certain extent, the model might have fair predictions for Google stock price even during the unprecedented time of the global pandemic. The attempt of building LSMT model is to experiment a well-acclaimed deep learning method for time series prediction. However, there should be more work on improving the LSTM model. Figure 6 shows the learning curve telling whether loss function is able to converge. It indicates that a subset of validation during the training process does not have its loss converging together with the training set. It means the model has yet to find the global or even local minimum. However, when increasing the number of epochs to help the model find its convergence,

paradoxically, the results were much less positive. Besides, comparisons between LSTM model with other machine learning models should be made for future study. Such task would help practitioners have more comprehensive and critical standpoints for an emerging method of LSTM in making historical dependency predictions.

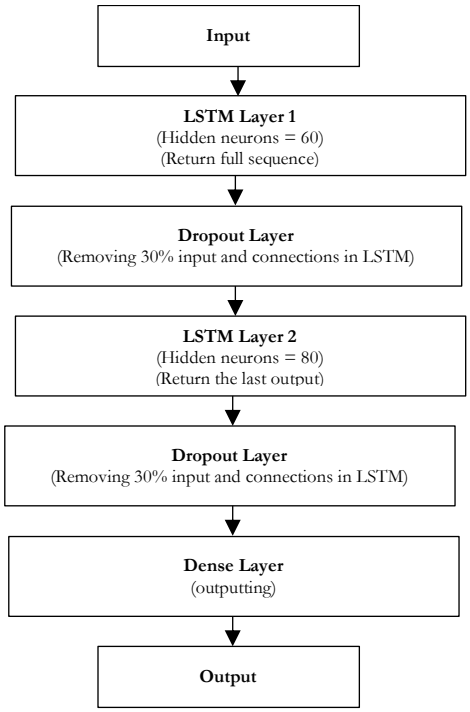


Figure 4. Final LSTM's architecture

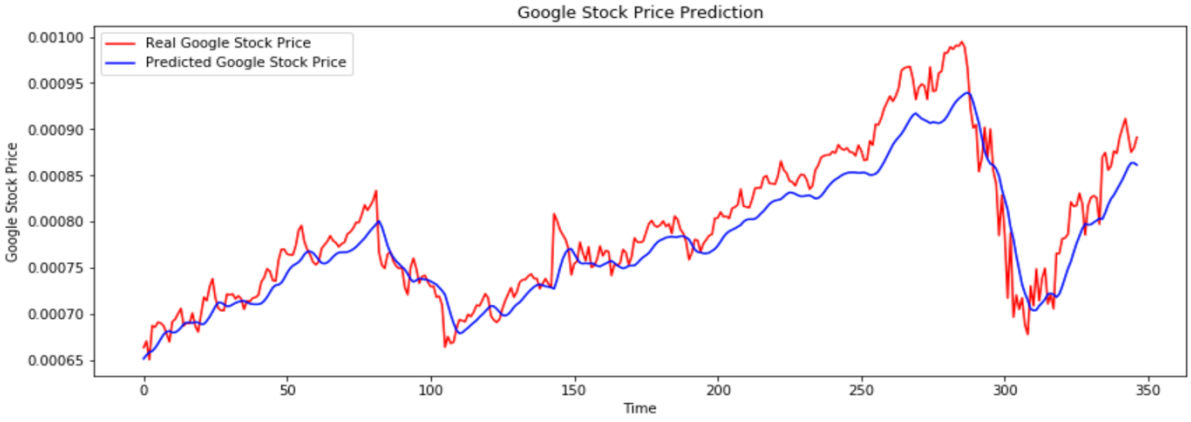


Figure 5. Stock Price Prediction

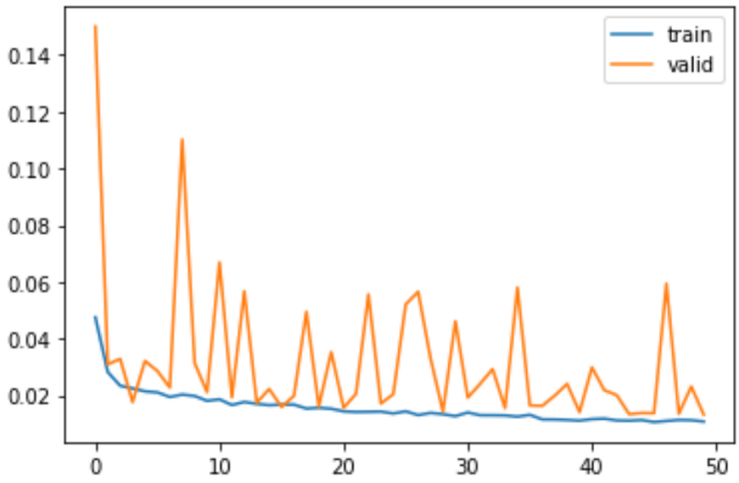
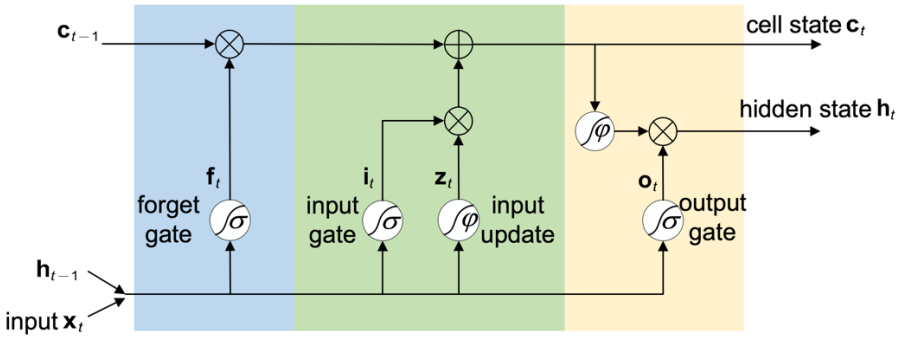


Figure 6. Train Loss vs Validation Loss during training process

Appendix

Appendix 1. How LSMT works

At a particular time t with the given hidden state vector h_{t-1} from the previous memory block and the input vector x_t. In the forget gate, the first step is to calculate how much information f_t is used from the previous memory cell (1) and how much information i_t is used from the input (2). Next, a cell input activation z_t is calculated (3). Then, the cell state vector is updated from c_{t-1} to c_t (4). The last step is to decide what to output which is the output vector of the LSTM unit h_t calculated from the output gate o_t (6).



	Where:
	$x_t \in \mathbb{R}^d$: input vector to the LSTM unit
	$f_t \in \mathbb{R}^h$: forget gate's activation vector
	$i_t \in \mathbb{R}^h$: input/update gate's activation vector
	$o_t \in \mathbb{R}^h$: output gate's activation vector
	$h_t \in \mathbb{R}^h$: hidden state vector also known as output vector of the LSTM unit
(1) $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$	$z_t \in \mathbb{R}^h$: cell input activation vector
(2) $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$	$c_t \in \mathbb{R}^h$: cell state vector
(3) $z_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$	$W \in \mathbb{R}^{h \times d}, U \in \mathbb{R}^{h \times d}$ & $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training.
(4) $c_t = f_t \circ c_{t-1} + i_t \circ z_t$	The superscripts d and h refer to the number of input features and number of hidden units, respectively.
(5) $o_t = \sigma(W_o x_t + U_0 h_{t-1} + b_0)$	The initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product. The subscript t indexes the time step.
(6) $h_t = o_t \circ \tanh(c_t)$	The sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$
	The tanh function $\tanh(x) = 2\sigma(2x) - 1$

LSTM equations

References

Finsveen, L., (2018). *Time-series predictions with Recurrent Neural Networks: Studying Recurrent Neural Networks predictions and comparing to state-of-the-art Sequential Monte Carlo methods*. Trondheim: Norwegian University of Science and Technology.

Gamboa, J. (2017). *Deep Learning for Time-Series Analysis*. arXiv:1701.01887v1 [cs.LG].

Hua Y., Zhao, Z., Li, R., Chen, X., Liu Z. & Zhang, H. (2019). *Deep Learning with Long Short-Term Memory for Time Series Prediction*. In: IEEE Communications Magazine, 57(6), pp.114-119, DOI: 10.1109/MCOM.2019.1800155.