Experiment 7 : Long Short-Term Memory (LSTM)

Tanmay Rathod

1 Introduction

Long short-term memory (LSTM) is a variation of an RNN model. An RNN can only memorize short-term information, but LSTM can handle long time-series data.

2 Aim

Develop an LSTM (Long Short-Term Memory) model to predict Google stock prices using historical data.

3 Objectives

To develop and deploy a robust LSTM (Long Short-Term Memory) model capable of accurately forecasting future Google stock prices based on historical data. This involves implementing a comprehensive data preprocessing pipeline to prepare the dataset for training. The model architecture will consist of multiple LSTM layers with dropout regularization to prevent overfitting and enhance

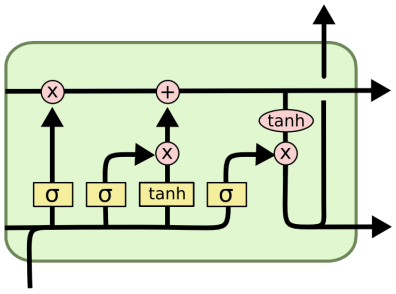

Figure 1: LSTM architecture

generalization. The primary objective is to train the LSTM model on a substantial portion of historical stock price data, ensuring it learns the intricate patterns and trends present in the dataset. Once trained, the model will be utilized to generate predictions for future stock prices, specifically targeting the year 2017 in this case. The final objective is to visualize the predicted stock prices alongside the actual values, enabling a thorough assessment of the model's performance and its ability to capture the underlying dynamics of the stock market. Through meticulous execution of these objectives, the aim is to deliver a reliable and accurate forecasting tool that can assist investors and analysts in making informed decisions regarding Google stock investments.

4 Methodology

4.1 Data Preprocessing

1. Import the libraries
2. Import the training set
3. Feature scaling
4. Creating a data structure
5. Reshaping the data

4.2 Building and Training the RNN

1. Importing the Keras libraries and packages
2. Initialising the RNN
3. Adding LSTM layers with Dropout regularisation
4. Adding the output layer
5. Compiling the RNN
6. Fitting the RNN to the training set

4.3 Making Predictions and Visualising Results

1. Getting the real stock price of 2017
2. Getting the predicted stock price of 2017
3. Visualising the results

5 Theory

The main distinction between RNN and LSTM designs is that the LSTM's buried layer is a gated unit or gated cell. It is made up of four layers that interact with one another to generate the cell's output as well as the cell's state. After that, these two items are passed on to the next concealed layer. LSTMs contain three logistic sigmoid gates and one tanh layer, unlike RNNs, which have only one neural net layer of tanh.

5.1

Forget Gate: The forget gate in an LSTM network is responsible for determining which information from the previous cell state C_{t-1} should be forgotten or retained for the current time step t. It takes as input the concatenation of the current input x_t and the previous hidden state h_{t-1}, and applies a sigmoid activation function to produce a vector of values between 0 and 1.

These values act as gates, controlling the flow of information from the previous cell state. A value of 1 indicates that the corresponding information should be retained, while a value of 0 indicates that it should be forgotten. Mathematically, the forget gate operation can be represented as:

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

Where:

- W_f are the weights associated with the forget gate,
- b_f is the bias term for the forget gate,
- σ represents the sigmoid activation function,
- $[h_{t-1}, x_t]$ denotes the concatenation of the previous hidden state and the current input, and
- f_t is the vector of forget gate values.

5.2

Input Gate: The input gate determines which new information should be stored in the cell state C_t. Similar to the forget gate, it takes the current input x_t and the previous hidden state h_{t-1} as inputs, and applies a sigmoid activation function to generate a vector of values between 0 and 1. Additionally, it applies a tanh activation function to generate a vector of candidate values that could be added to the cell state.

The input gate then combines these two sets of values to produce an update to the cell state. Mathematically, the input gate operation can be represented

as:

$$
\begin{aligned}
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\end{aligned}
$$

Where:

- W_i, W_C are the weights associated with the input gate and the cell state update respectively,
- b_i, b_C are the bias terms for the input gate and the cell state update respectively,
- $\tilde{C}_t$ is the vector of candidate values for the new cell state,
- i_t is the vector of input gate values, and
- C_t is the updated cell state.

5.3

Output Gate: The output gate determines the information to be output at the current time step t, which is based on the current input x_t and the previous hidden state h_{t-1}, as well as the updated cell state C_t. It applies a sigmoid activation function to the concatenation of these inputs to generate a vector of values between 0 and 1.

This gate then regulates which parts of the cell state should be output as the hidden state h_t, by applying the tanh function to the updated cell state C_t and multiplying it element-wise with the output gate values. Mathematically, the output gate operation can be represented as:

$$
\begin{aligned}
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned}
$$

Where:

- W_o are the weights associated with the output gate,
- b_o is the bias term for the output gate,
- o_t is the vector of output gate values, and
- h_t is the hidden state output at time step t.

6 Results

The loss values for the training epochs are as follows:

```
Epoch 100/100 - loss: 0.0016
```

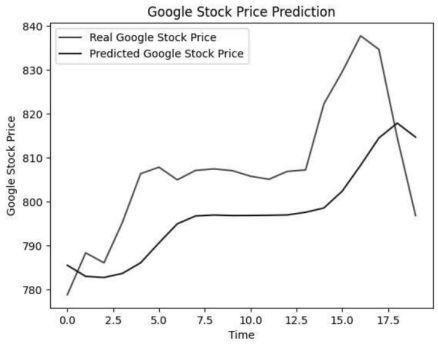

Figure 2: Enter Caption

7 Conclusion

The model is used to predict the stock prices for the test set, and the results are then compared to the actual stock prices. Finally, a visualization is generated to illustrate the predicted and actual stock prices for 2017.

The LSTM model demonstrates an ability to capture temporal dependencies in the data and make reasonably accurate predictions, as depicted in the visualization. Further optimization and fine-tuning of the model parameters could potentially enhance its performance.