LNMIIT
The LNM Institute of Information Technology

LSTM: Stock Market Prediction

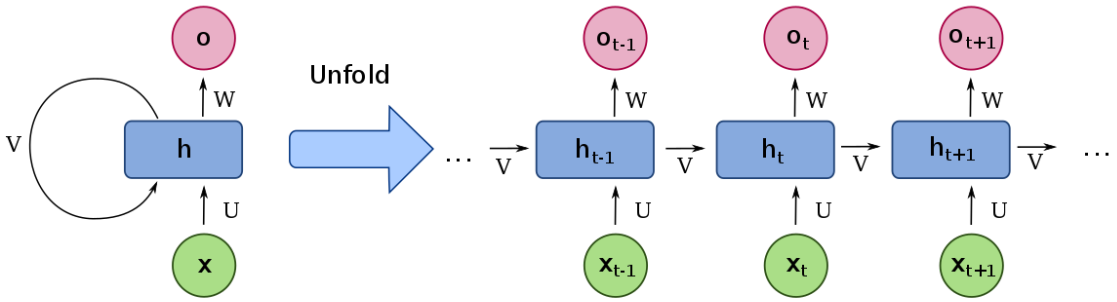
Coding algorithm, development, and execution

Overview

In this project we have used the concepts of **LSTM**(Long short term memory) and **RNN**(Recurrent Neural Networks) and tried to propose a model that predicts the opening price for a stock using the LSTM algorithm.

Algorithms used

RNN: A **recurrent neural network** (**RNN**) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

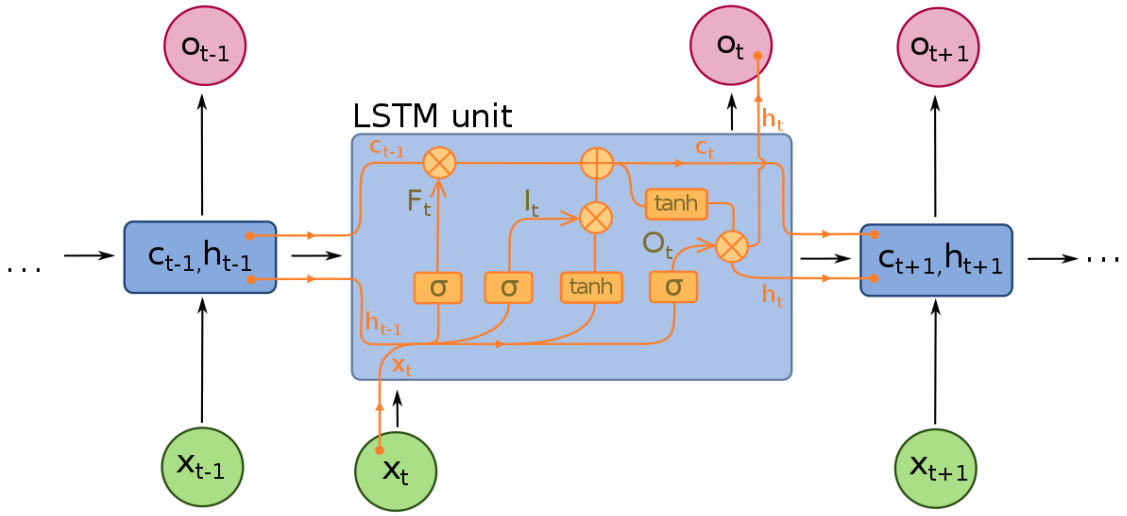


The term “recurrent neural network” is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite

impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

LSTM: **Long short-term memory** (**LSTM**) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.



In theory, classic (or "vanilla") RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem with vanilla RNNs is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can

"vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

Development

The development of the model was initiated by learning more about LSTM and RNN. This was achieved through reading of various articles and videos. One of the major contributing sources were YouTube videos of MIT Introduction to deep learning. (https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI)

Also for the dataset various sources were taken into consideration and finally Yahoo Finance was chosen.(https://in.finance.yahoo.com/) Various stocks were taken into consideration to train the model upon and finally Google(GOOG) stock was chosen. (https://in.finance.yahoo.com/quote/GOOG?p=GOOG)

Also we gained some knowledge about the Memory Network through the links provided.(https://paperswithcode.com/method/memory-network , https://arxiv.org/pdf/1410.3916v11.pdf)

Links of Articles:

https://en.wikipedia.org/wiki/Recurrent_neural_network

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

https://en.wikipedia.org/wiki/Long_short-term_memory

https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://keras.io/api/layers/recurrent_layers/lstm/\

Execution

The language used to execute the model is Python because of its ease to work in and libraries supporting the use of various algorithms and layers(such as MinMaxScaler, Sequential, etc.). For this the team members needed to polish their python skills and various resources were used.(Eg: https://www.tutorialspoint.com/python/index.htm)

The Execution was initially thought to be conducted on the Jupyter notebooks(for their ease to use) but it was hard to collaboratively work on that platform so then we migrated the code to Google collab.
(https://colab.research.google.com/drive/1JQWRwMvq8pG2t72Q-CFoD6zn0ur7Bil2?authuser=1
)

Link of Jupyter Notebook and Downloaded DataSet:

https://github.com/samyakgoyal/Stock-Prediction-Using-LSTM

The dataset was first downloaded from Yahoo Finance and then loaded.

After loading and scaling the dataset(using MinMaxScaler from sklearn.preprocessing) the next step was to build the LSTM model.

For this we imported Sequential(type of model) , Dense(layer required at end of the model to summarize), LSTM(layer to implement LSTM), Dropout(layer to avoid overfitting) from **Keras TensorFlow.**

Then the model was tested for various parameters of the model to tune the model for better performance through the graph plot at the end of the code.

Also the error was calculated and the graph for that was also plotted for better visualization.

The Code-Flow was as Follow:

Loading DataSet -> Feature Scaling -> Creating Data With TimeStamps -> Building the LSTM model -> Graph generation of predicted and actual values and error.