Application of LSTM to Stock Index Forecasting

Computational Project Report
Managerial Forecasting and Decision Analysis (Fall 2023-EIND 468-588)

Hans Petersen
Mechanical & Industrial Engineering Department
Montana State University
Bozeman, Montana 59717
Email: hans.c.petersen2001@gmail.com

Merlin McKean
Mechanical & Industrial Engineering Department
Montana State University
Bozeman, Montana 59717
Email: merlinmckean@gmail.com

Abstract

In the realm of financial forecasting, the demand for accurate and efficient predictive models has intensified, particularly in the context of stock price prediction. Machine and deep learning-based algorithms have emerged as powerful tools for addressing prediction challenges in time series analysis, showcasing superior performance compared to traditional regression-based models. This study focuses on the application of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) models, in the context of stock index prediction. LSTM, characterized by its memory-retaining "gates," demonstrates significant superiority over Autoregressive Integrated Moving Average (ARIMA) models. The central question of interest is whether the inherent gates within LSTM already offer accurate predictions or if additional training could further enhance its performance.

Conducting a comprehensive behavioral analysis and comparison of LSTM models, this paper investigates the extent to which additional training layers improve parameter tuning. The findings did not necessarily reveal that LSTM, with its intrinsic gates, provides robust predictions compared to ARIMA models. We understand that traditionally LSTM models do out perform ARIMA models however we were not able to show this with our data. Nevertheless, the study contributes valuable insights into the efficacy of LSTM models in capturing intricate patterns within financial time series data, paving the way for enhanced stock index prediction using deep learning approaches.

KEYWORDS
forecasting, stock price prediction, LSTM, BiLSTM, neural network

Literature Review

There have been several past papers regarding LSTM models, their BiLSTM extensions, and the potential applications of both. One paper, regarded as seminal in the field, is titled “Long Short-Term Memory” in which Hochreiter and Schmidhuber were the first to introduce a new type of artificial neural network architecture, the LSTM model. The main focus of the paper was to address the challenges of learning and remembering long-term dependencies in sequential data, which were difficult for traditional recurrent neural networks (RNNs) to capture. The key contributions of the paper include an introduction

of LSTM architecture, memory cells, gating mechanisms, training LSTMs, and preliminary experimental results. This paper and the LSTM model it presents have become cornerstones in the artificial learning community.

Second, in their work "Bidirectional Recurrent Neural Networks", Schuster and Paliwal explain first how the RNN can be extended to a bidirectional RNN (BRNN). They then detail how the model can be trained simultaneously in a positive and negative time direction, a characteristic which generally gives better results than other models, including regular RNNs.

Lastly, we aim to use the work of our reference paper, Siami-Namini et al., titled "The Performance of LSTM and BiLSTM in Forecasting Time Series." This paper aimed to analyze, understand, and implement LSTM and BiLSTM models with the goal of predicting future values of various stock indices. It describes the issues discussed below regarding traditional modeling techniques such as ARIMA and its variations, which are suitable for short-term forecasting but begin to lose accuracy on longer time horizons. The paper then elaborates on the definitions of and differences between LSTM and BiLSTM models, as our paper will do in subsequent sections.

Siamin-Namini et al. ask the following research questions:

1) Does the binary nature of the BiLSTM model (data passed through past-to-future and future-to-past) make an improvement on LSTM's prediction rates?
2) How is input data treated differently between these two models?
3) Which architecture reaches equilibrium faster?

The paper also explains the issue plaguing many RNN researchers - the vanishing gradient problem - which, as we explain below, is a large factor in traditional RNNs' failure in capturing long-term dependencies.

The bulk of the work of Siamin-Namini et al. appears in their "LSTM vs. BiLSTM: An Experimental Study" section. Here, they first explain the dataset used. It has been split into a 70% training set and a 30% test set, with a total of over 120,000 observations of the historical daily, weekly, and monthly closing prices of various stock exchanges (N225, IXIC, HSI, GSPC, and DJI) as well as a daily closing price for IBM's stock. They continue to detail assessment metrics used, specifically Root Mean Squared Error (RMSE), which measures the distance between actual and predicted values generated by the model.

Following the assessment metrics, the general and technical aspects of both LSTM and BiLSTM models were expounded upon, including in-depth explanations about their underlying mechanisms. Specifically, a difference between Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs) is outlined. ANNs models travel directly from input to output without using any feedback from data that has already been trained. This means the ANN model has no memory, and therefore one layer's output has no effect on the training of the same layer. These memoryless neural networks are commonly used in tasks like image and pattern recognition, classification, and regression where the input data is independent of each other. RNNs, on the other hand, have connections that form a directed cycle, allowing information to be stored and processed over time. This cyclic structure enables RNNs to handle sequential data and capture temporal dependencies. This process is called feedback. However, RNNs present a problem in which feedback has no preservation ability, which means that long inputs are forgotten. Long inputs refers to a sequence of data with a large number of elements or time steps. For example, in natural language processing, processing a long sentence or document could be considered

dealing with a long input sequence. One potential solution to this issue is presented as LSTMs and BiLSTMs, though we wait until the subsequent sections to provide a thorough review of the underlying mechanics of both models.

After applying both models to the over 120,000 data points, a Root Mean Square Error (RMSE) was calculated for each index/frequency combination. The results from Siamin-Namini et al. show that both the LSTM and BiLSTM algorithms performed significantly better than an ARIMA model, while the BiLSTM variant consistently showed slightly better results than its counterpart, LSTM. As displayed in a summary table, the LSTM algorithm performed 88.07% better than ARIMA on average (RMSE values of 39.09 and 302.96, respectively) while the BiLSTM performed 48.40% better than LSTM, which had an average RMSE of 20.17. These average patterns hold when we inspect the models' performances on monthly Dow Jones Industrial Average closing prices; we observe RMSEs of 516.97, 69.53, and 23.69 for ARIMA, LSTM, and BiLSTM, respectively, which again show that using a BiLSTM model proves more accurate when forecasting, all else equal - particularly computational efficiency.

Problem Description

Forecasting the price of a stock index is a real-world problem that falls under the broader category of time series prediction. Time series forecasting involves predicting future values based on past observations, and in the context of stock prices, it's a complex task due to the presence of various factors influencing market dynamics. Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) models have been widely used for such tasks, as they can capture temporal dependencies and patterns in sequential data.

The problem we aim to address is predicting the future values of a stock index, specifically, the Dow Jones Industrial Average. Stock prices are inherently sequential, and their movements are influenced by a multitude of factors, including economic indicators, company performance, geopolitical events, and market sentiment. The objective is to build a predictive model that can leverage historical stock prices and other relevant features to forecast future values accurately. To achieve this, historical stock price data was collected. This dataset typically includes daily or hourly closing stock prices along with other relevant information such as trading volume, economic indicators, and news sentiment scores. The dataset is split into training and testing sets, with the training set used to train the model and the testing set used to evaluate its performance.

However, an underlying problem exists in the process of solving the above real-world problem, called the vanishing gradient problem. According to Van Otten (2023), the vanishing gradient problem is a common issue in the effective implementation of RNNs and the training of deep learning models. The problem occurs when the gradients, which are the loss function's rate of change, become too small for the purpose of updating and improving the model. During backpropagation, which is the process of feeding error rates back into the model so it can learn how to increase its accuracy, miniscule gradients make it difficult to update each parameter and improve the model. This issue is particularly relevant in RNNs, where the small gradients can lead to non-convergence. This translates to difficulty in effectively learning from the raw data and capturing long-term dependencies in sequential data. Alleviating the vanishing gradient problem is crucial in using RNNs to solve problems such as time series forecasting.

Mathematical Modeling

FIGURE 1

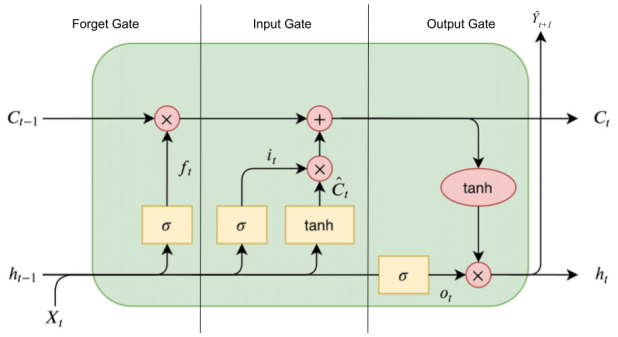


Figure 1 represents the data manipulation process our LSTM model used to train itself and forecast future observations.

1) Cell State and Hidden State:
    a) The core of an LSTM is its cell state, which acts as a memory unit. It runs through the entire sequence and can capture information over long distances. The hidden state is the output at a given time step. The process of forecasting predictions, or generating $\hat{Y}_t$ observations, is the process of "converting" the cell state to the hidden state and is represented by the path of the black arrows.

2) Gates:
    a) LSTMs use three gates to control the flow of information: Forget Gate, Input Gate, and Output Gate. These gates are neural network layers that determine what information should be discarded, what new information should be stored, and what information should be output.

3) Forget Gate:
    a) The Forget Gate decides which information from the cell state needs to be discarded or kept. It takes the previous hidden state h_{t-1} and the current input x_t and produces a value between 0 and 1 for each element in the cell state C_{t-1}. This is achieved through a sigmoid activation function: $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$ where f_t is the output of the Forget Gate at time t. The Forget Gate determines what information should be discarded from the cell state. In the context of stock index prediction, f_t helps the model decide which historical information is less relevant for predicting the current price.

4) Input Gate:
    a) The Input Gate determines what new information should be stored in the cell state. It consists of two parts: a sigmoid layer that decides which values to update and a tanh layer that creates a vector of new candidate values by utilizing the following formulas:
    $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$

Where i_t is the input gate output at time t. The input gate decides what new information should be added to the cell state. In stock price prediction, i_t helps the model identify which features or patterns in the current input (x_t) are important in updating the cell state.
$\hat{C}_t = tanh(W_c * [h_{t-1}, x_t] + b_C)$
Where $\hat{C}_t$ is the candidate cell state, which represents the new information that the model is considering adding to the cell state. In our context $\hat{C}_t$ could contain information about potential trends or patterns in stock price. The model would look backward at the trends or patterns, which may come in the form of company performance, overall sentiment, or any number of catalyst events. The correlation of these events on the index's price would be factored and a candidate $\hat{C}_t$ may be deemed an updated (accepted) cell state.

5) Updating the Cell State:
    a) The cell state is updated by combining the information retained from the previous cell state (as determined by the forget gate) and the new candidate values (as determined by the input gate).
    $C_t = f_t * C_{t-1} + i_t * \hat{C}_t$ where C_t is the updated cell state at time t. It is a combination of the previous cell state (C_{t-1}) that the model decided to retain and the new information ($\hat{C}_t$) that the model decided to add. C_t serves as the long-term memory aspect of the LSTM, which is largely responsible for this algorithm's dominance over traditional models such as ARIMA.

6) Output Gate:
    a) The Output Gate determines the next hidden state based on the updated cell state. It uses the previous hidden state and the current input to produce a filtered version of the cell state.
    $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$
    Where o_t is the output of the output gate at time t. The output gate determines the next hidden state (h_t) based on the updated cell state (C_t). In the context of predicting stock prices, o_t helps the model produce the final prediction based on the relevant information stored in the cell state.
    $h_t = o_t * tanh(C_t)$
    Where h_t is the hidden state at time t. This represents the output of the LSTM for the current time step. h_t contains information that the model has deemed relevant for making predictions based on the historical input sequence.

BiLSTMs extend this approach by processing the stock index data in both forward and backward directions, though they are not central to our work. Each time step in the sequence considers not only past information but also future information, providing a more comprehensive context for prediction. In the context of stock indices, this bidirectional processing allows the model to capture dependencies influenced by both preceding and succeeding market conditions. The final hidden state in a BiLSTM is typically a concatenation of the forward and backward hidden states, offering a more nuanced representation of the temporal dynamics in stock index movements. This bidirectional nature is advantageous for capturing intricate patterns and trends that might be overlooked in a unidirectional approach.

Data

With a goal of applying our own LSTM model to the same data used in the work of Siamin-Namini et al., and considering that most stock indices had the same relative results in terms of RMSE, only one stock index was chosen for our analysis. We chose the closing prices of the Dow Jones Industrial Average ($DJI) on daily, weekly, and monthly time horizons, which were available on Yahoo Finance. It's important to note that because $DJI is an index fund acting as an average of hundreds of individual stocks, its behavior will be inherently less volatile than that of individual stocks. This should make the index less susceptible to large shocks that would otherwise be difficult to forecast.

Model Architecture

The LSTM model developed to approximate Siami-Namini et al.'s model focused on four main parameters to dictate the structure: neurons, time steps, batch size, and epochs. The number of neurons in an LSTM layer determines its capacity to learn from the data. Each gate has a set of neurons weighing the output to process and dictate which data gets stored in the long- and short-term memory. A larger number of neurons in the model implies a dataset with more intricacies for the model to capture and use as inputs. However, too many neurons will cause overfitting of the model to the training data. The value of each neuron is adjusted when each training data point is passed into the model. The actual value of each neuron is not important to us as the model builders, but they function as the brain of the model, allowing it to learn. The other parameters are simpler, time steps is the number of previous observations being fed into the model. The larger the time scale allows the model to predict y values using more information, this can also lead to overfitting and greatly increases the computational time. Batch size is the number of steps the model takes before updating the neuron structure. A larger batch size will decrease the computational cost of training the model, decreasing the amount of small variation the model captures. When looking at series like a stock index price, there are many small shocks caused by external forces that we would prefer the model not to adjust for. These three parameters are referred to as hyperparameters, they govern how the model learns and choosing a reasonable set is necessary for any machine learning model. There are more hyperparameters that can be used, however they were not mentioned in the analysis of Siami-Namini et al**.** Epochs refer to the number of times the data is passed through the model, allowing the model to refine its identification of trends. Again, this can lead to overfitting if the model is given enough time to approximate the data too closely.

In the context of LSTM networks, hyperparameters are the external configuration settings that are not learned from the training data but are set before the training process begins. These parameters are crucial as they influence the structure and performance of the LSTM model. In the context of the LSTM under analysis, the relevant hyperparameters are neurons, time steps, and batch size. The number of epochs, while important in dictating how many times to iterate through the gates, is not considered a hyperparameter because it has no effect on the algorithm's underlying structure. As for neurons, time steps, and batch size, the nature of RNNs and LSTM by extension makes it near impossible to select the optimal hyperparameters by hand. To choose a suitable time step for our model, we started with a large window and narrowed it down according to a seemingly balanced tradeoff between computational time and accuracy, represented by runtime and RMSE, respectively. This process resulted in a set of four time steps, [12, 32, 64, 128], five batch sizes [4,8,16,21,64] and a range of [1, 300] neurons. This means that

for each model predicting daily, weekly, and monthly DJI prices, each dataset was sequenced into 12ths, 32nds, etc. with the number of nodes being set to, on average, one neuron.

Results and Conclusion

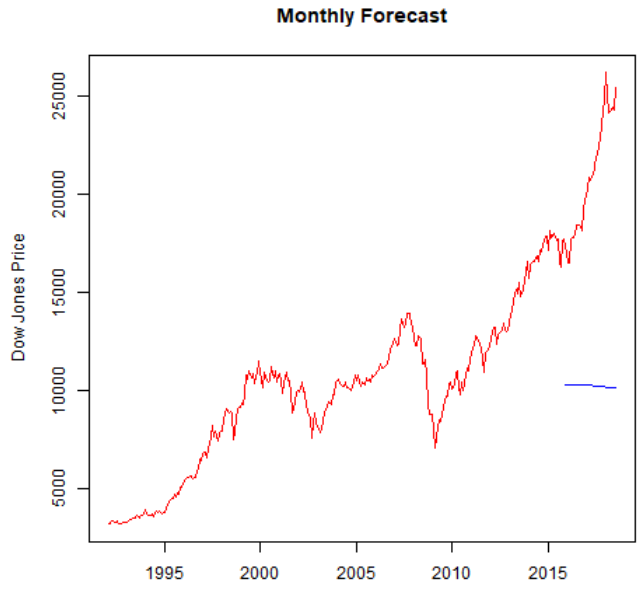
After applying our revised LSTM network to the daily, weekly, and monthly DJI data, we were able to create a forecast, from which we can now make a conclusion. Table 1 displays these results according to the number of neurons, batch size, and time step that allows for the best (lowest) RMSE for each frequency of data, and is visualized in Graphs 1,2, and 3. Unfortunately, despite countless tweaks, iterative optimization and hours of computing time, we were unable to arrive at anything reasonably close to the RMSE achieved by Siami-Namini et al., much less an improvement. With average RMSEs above 8,000 for the monthly and weekly datasets and a quite large RMSE of over 11,000 for the daily data, our results were disappointingly different from our reference model.

Although we tried to replicate everything not inherent to the model (common data, “adam” optimizer, same accuracy metrics, same test/train split), the difference in results points to a discrepancy in the model architecture, i.e. a stark difference in the underlying code. Because the authors provided only pseudo-code as opposed to line-by-line functions and dependencies, there are countless possibilities in terms of where we could have erred. For example, a primary difference to be drawn between our model and our reference paper’s model is access to computational resources. With presumably significantly more powerful computing power and time to train the model, Siami-Namini et al. would have had the freedom to choose any number of neurons and layers, which were our limiting factors. However, the fact that their hyperparameter values remain unknown makes this difficult to verify.

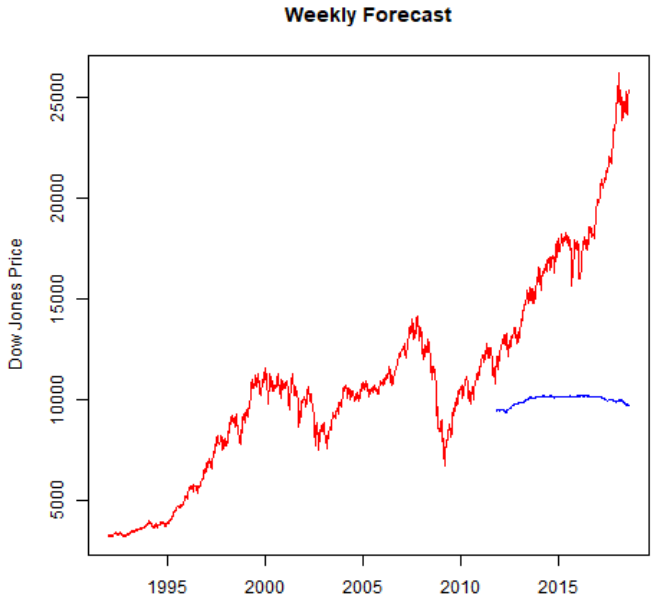
Despite a lack of success in improving on the authors’ results, this was a valuable project in many ways. First, we would not have otherwise been exposed to recurrent neural networks such as LSTMs nor seen their wide-ranging applications, in the realm of stock price prediction and beyond. In a world in which machine learning and algorithms will continue to be integrated into more parts of our personal, occupational, and academic lives, being familiar with an LSTM is merely scratching the surface of opportunity.

Finally, there’s great value in defining the architecture and tuning of any such models, particularly in analyzing them with scrutiny and critique. If we were forecasting students then this could serve as the base for a capstone project, and the sheer amount of skills learned from this project alone are staggering. The entire forecasting process, from choosing a research paper with a corresponding model, analyzing it, and attempting to replicate its results using our own revised model provided many transferable skills for the workforce and beyond.

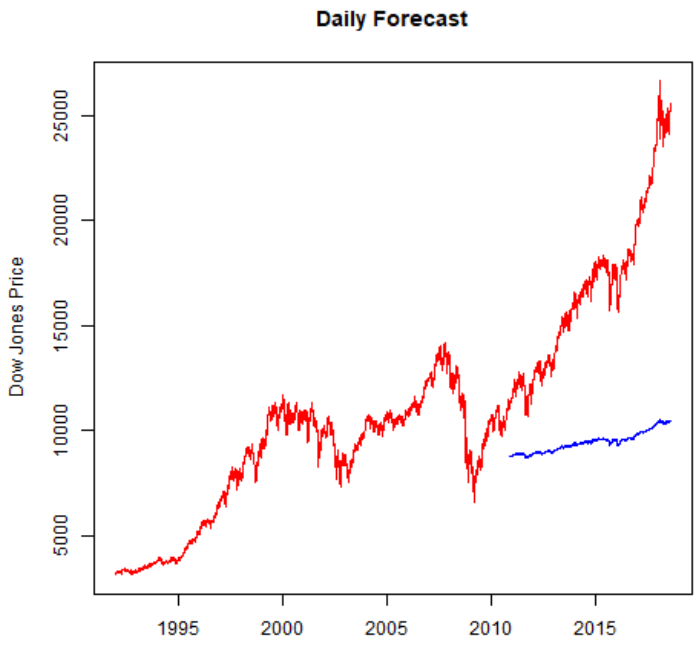
Index



Graph 1



Graph 2



Graph 3

TABLE 1: RMSEs of LSTM on Monthly, Weekly, & Daily DJI Data

	Neurons	Batch	Time	RMSE
Monthly	1	6	64	8123.995
Weekly	1	6	64	8434.099
Daily	1	6	64	11171.363

References

Hochreiter, Sepp, and Jürgen Schmidhuber. “Long Short-Term Memory.” Neural Computation, vol. 9, no. 8, 1997, pp. 1735–1780.

Mar Ingolfsson, T. (2021, November 10). *Insights into LSTM architecture*. Thorir Mar Ingolfsson. https://thorirmar.com/post/insight_into_lstm/

Schuster, Mike, and Kuldip K. Paliwal. “Bidirectional Recurrent Neural Networks. IEEE Journals and Magazine. IEEE Transactions On Signal Processing, IEEE, 11 Nov. 1997, ieeexplore.ieee.org/document/650093.

Siami-Namini, Sima, et al. “The Performance of LSTM and BiLSTM in Forecasting Time Series.” NSF Public Access Repository, IEEE International Conference on Big Data, 2019, par.nsf.gov/servlets/purl/10186554.