Time Series Forecasting: ARIMA vs. LSTM

Emily Nomura EMILY_NOMURA@ALUMNI.BROWN.EDU
Data Science Initiative
Brown University
Machine Learning: from Theory to Algorithms
DATA 2060, Spring 2023

Abstract

This paper provides a theoretical introduction to the Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) models. It compares the two models with respect to the minimization of their error rates and discusses successful applications of LSTM for time series forecasting. This paper summarizes the major findings of *A Comparison of ARIMA and LSTM in Forecasting Time Series*, which was presented at the 2018 17th IEEE International Conference on Machine Learning and Applications (Siami-Namini et al., 2018).

Keywords: Time Series Forecasting, Deep learning, Autoregressive Integrated Moving Average (ARIMA), Long Short-Term Memory (LSTM)

1. Introduction

Time series forecasting is applicable in many sectors. It is most notably used in business and finance, as the U.S. stock market is incredibly volatile. However, it also has relevance in geology (earthquake prediction), environmental science (global temperature models), and public health (disease modeling).

Given the "big boom" of data collection in the past few decades, time series data is not difficult to acquire. Time series forecasting is applicable to any type of business, and the outcomes of interest can vary greatly in order to fit the business's needs.

When working with time series data, the most classic technique is often some type of autoregressive model, the most popular of which is the Autoregressive Integrated Moving Average (ARIMA) model. The mathematics and pseudocode for ARIMA is discussed further in Section 2.

Recent advances in machine learning and deep learning techniques have led to their widespread implementation in various business problems, including time series forecasting. Long Short-Term Memory (LSTM) models are reported to outperform typical time series models like ARIMA, with an average error rate reduction of 84%-87% (Siami-Namini et al., 2018). This paper will compare ARIMA and LSTM models with respect to the minimization of their error rates as implemented by Siami-Namini et al.

2. Autoregressive Integrated Moving Average (ARIMA)

An autoregressive model contains previous values of the target variable as predictors. The lagged values can be used directly in the model, however, it is more common to incorporate differencing, which uses the difference between the lagged values as predictors. Differencing "stabilizes" the predictors and ensures that the mean variance is stationary; a general rule of thumb in order to ensure a robust model (Yiu). The benefits of differencing can be observed when comparing the variation in Figure 1 to the stationarity of Figure 2.

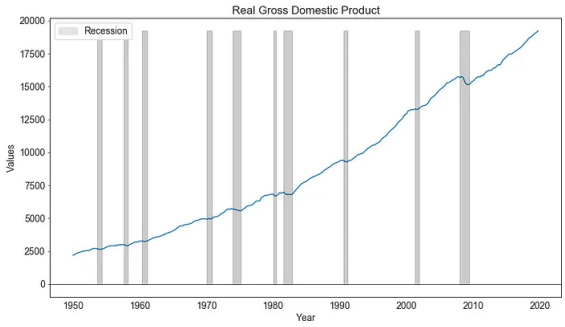

Figure 1: U.S. Real GDP (source: Federal Reserve Bank of St. Louis) (Yiu)

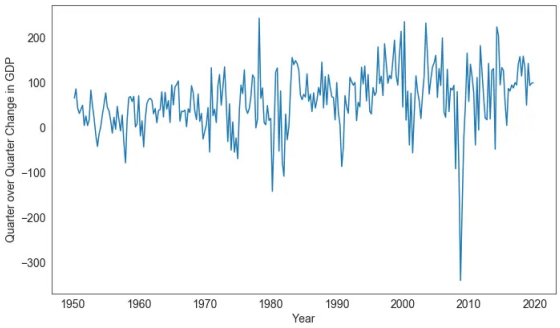

Figure 2: Change in Real GDP is more stationary (Yiu).

A moving average model uses the errors of past forecasts in order to make predictions (Binhuraib). This process is very similar to loss minimization, as the model is technically "learning" from its previous errors.

The ARIMA model combines the autoregressive model and moving average model. It incorporates differencing to forecast future values from past observations. Although an ARIMA model works well in many cases, Long Short Term Memory (LSTM) models have certain benefits over ARIMA for time series forecasting, and have gained popularity in recent years.

2.1 Mathematics

An autoregressive model of order p and a moving average model of order q can be written as in Equations (1) and (2).

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-1} + \epsilon_t \tag{1}$$

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \epsilon_{t-i} \tag{2}$$

When combined, an ARIMA model of order (p, q) can be written in the following form:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t + \sum_{i=0}^{q} \theta_i \epsilon_{t-1} \tag{3}$$

In the ARIMA equation, ϕ_i represents the autocorrelation coefficients and θ_i represents the model weights. These weights are "applied to the current and prior values of a stochastic term in the time series" (Siami-Namini et al., 2018).

In order to estimate a seasonal ARIMA model, the parameters p, q, and d must be solved for. p represents the number of lagged values the model takes in or the order of the autoregressive term. q represents the number of previous forecast errors that the model considers or the order of the moving average term. Finally, d stands for the order of differencing in the model (Binhuraib). The general form of an ARIMA model is denoted $ARIMA(p, d, q)$.

2.2 Pseudocode

In *A Comparison of ARIMA and LSTM in Forecasting Time Series*, the authors implement a rolling ARIMA algorithm that "performs multi-step out-of-sample forecast with re-estimation, i.e., each time the model is re-fitted to build the best estimation model" (Siami-Namini et al., 2018). They fit an $ARIMA(5, 1, 0)$ model initially before iteratively re-fitting the model in order to improve upon the baseline.

```
# Rolling ARIMA
Inputs: series
Outputs: RMSE of the forecasted data
# Split data into:
# 70% training and 30% testing data
1. size=length(series) * 0.70
2. train=series[0...size]
3. test=series[size...length(size)]
# Data structure preparation
4. history=train
5. predictions=empty
# Forecast
6. for each t in range(length(test)) do
7.       model=ARIMA(history, order=(5, 1, 0))
```

```
8.        model_fit=model.fit()
9.        hat=model_fit.forecast()
10.           predictions.append(hat)
11.           observed=test[t]
12.           history.append(observed)
13. end for
14. MSE = mean_squared_error(test, predictions)
15. RMSE = sqrt(MSE)
16. Return RMSE
```

3. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) models have significant importance in the time series forecasting world due to their ability to "remember" long-term seasonality and trends of time series data. In fact, Joos Korstanje, author of *Advanced Forecasting with Python*, states (Keith),

> The LSTM cell adds long-term memory in an even more performant way because it allows even more parameters to be learned. This makes it the most powerful [Recurrent Neural Network] to do forecasting, especially when you have a longer-term trend in your data. LSTMs are one of the state-of-the-art models for forecasting at the moment.

LSTM models are a type of deep learning model classified as a Recurrent Neural Network (RNN). The main purpose of an RNN is to predict the next step in the sequence of observations with respect to the previous steps observed in the sequence, similar to the goal of an ARIMA model (Siami-Namini et al., 2018).

The main difference between LSTM and ARIMA is the fact that LSTM is a type of neural network that consists of different "layers" used for various learning purposes. The most basic neural network consists of three primary layer types:

1. An input layer
2. Hidden layers
3. An output layer

The hidden layers in an LSTM model allow it to store a great deal of information from earlier stages, usually in a sequential order, with the goal of forecasting future trends. LSTMs are composed of interconnected "cells" or "modules" that are used to transport and store data.

Much like a conveyor belt line in a shipping facility, LSTMs contain "gates" where data can be disposed, filtered, or added when it is transported to the next cell in the sequence. There are three main types of gates in an LSTM (Siami-Namini et al., 2018):

- Forget gate. This takes on a value between 0 and 1 via a sigmoid activation function where 0 tells the model to completely ignore the data and 1 tells the model to completely remember/keep the data. The outputted values are point-wise multiplied by

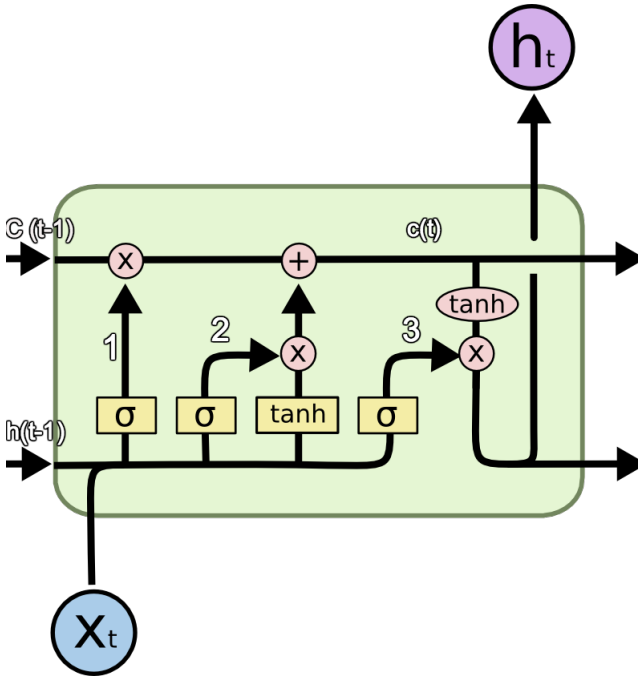

Figure 3: A LSTM Unit (Source: Understanding LSTM Networks.

the previous cell state in order to determine which data should be deemed irrelevant versus relevant.

- Memory or input gate. This gate acts as a filter for what data should be kept. It can be described as a point-wise addition of the point-wise multiplication of two components - the output of the sigmoid layer and the output of the hyperbolic tangent layer (Dolphin).

- Output gate. The decisions about the outputted values of each cell are made here using a sigmoid activation function. The output gate effectively "decides" the new hidden state. The values are based on the cell state and filtered/added data.

3.1 Pseudocode

Siami-Namini et al. (2018) developed a "multi-step univariate forecast algorithm" to implement a LSTM model for time series data. Keras and Theano packages in Python were used on a high performance computing cluster. The "ADAM" optimization algorithm was chosen and mean squared error was specified as the loss function.

```
# Rolling LSTM
Inputs: Time series
Outputs: RMSE of the forecasted data
# Split data into:
# 70% training and 30% testing data
```

```
1. size = length(series) * 0.70
2. train = series[0...size]
3. test = series[size...length(size)]
# Set the random seed to a fixed value
4. set random.seed(7)

# Fit an LSTM model to training data
Procedure fit_lstm(train, epoch, neurons)
5. X = train
6. y = train - X
7. model = Sequential()
8. model.add(LSTM(neurons), stateful=True))
9. model.compile(loss='mean_squared_error', optimizer='adam')
10. for each i in range(epoch) do
11.       model.fit(X, y, epochs=1, shuffle=False)
12.       model.reset_states()
13. end for
return model

# Make a one-step forecast
Procedure forecast_lstm(model, X)
14. yhat = model.predict(X)
return yhat

15. epoch = 1
16. neurons = 4
17. predictions = empty
# Fit the lstm model
18. lstm_model = fit_lstm(train,epoch,neurons)
# Forecast the training dataset
19. lstm_model.predict(train)

# Walk-forward validation on the test data
20. for each i in range(length(test)) do
21.    # make one-step forecast
22.    X = test[i]
23.    yhat = forecast_lstm(lstm_model, X)
24.    # record forecast
25.    predictions.append(yhat)
26.    expected = test[i]
27. end for
28. MSE = mean_squared_error(expected, predictions)
29. Return (RMSE = sqrt(MSE))
```

4. Successful Applications: Siami-Namini et al. (2018)

In their comparison of ARIMA vs. LSTM, Siami-Namini et al. (2018) sought to answer the research question: "Which algorithm, ARIMA or LSTM, performs more accurate prediction of time series data?"

4.1 Dataset and Data Preparation

Siami-Namini et al. (2018) used monthly financial time series data from the Yahoo finance website between January of 1985 and August of 2018. The dataset included Nikkei 225 index (N225), NASDAQ composite index (IXIC), Hang Seng Index (HIS), S&P 500 commodity price index (GSPC), and Dow Jones industrial average index (DJ) (Siami-Namini et al., 2018). The analysis also included another dataset using economic time series data from the Federal Reserve Bank of St. Louis and the International Monetary Fund (IMF) Website.

Stock	Observations		Total
	Train 70%	Test 30%	
N225	283	120	403
IXIC	391	167	558
HSI	258	110	368
GSPC	568	243	811
DJI-Monthly	274	117	391
DJI-Weekly	1,189	509	1,698
MC	593	254	847
HO	425	181	606
ER	375	160	535
FB	425	181	606
MS	492	210	702
TR	593	254	847

Figure 4: The number of time series observations (Siami-Namini et al., 2018).

The "Adjusted Close" variable was chosen as the only financial time series feature to be utilized in the ARIMA and LSTM models. The data was split into 70% training and 30% testing.

4.2 Assessment Metric

There are many different evaluation metrics that may be used for time series forecasting. Siami-Namini et al. (2018) chose to use Root-Mean-Square Error (RMSE).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2} \tag{4}$$

N represents the total number of observations in the data, x_i is the true value, and $\hat{x_i}$ is the predicted value.

4.3 Results and Discussion

The results of Siami-Namini et al. (2018) are reported in Figure 5. For the stock market data, the average RMSE of the Rolling ARIMA model was 511.481 and the average RMSE of the Rolling LSTM model was 64.213. Thus, LSTM achieved an average 87.445 of reductions in error rates compared to ARIMA. For the economic (non-stock market) data, the average RMSE of the Rolling ARIMA model was 5.999 and the average RMSE of the Rolling LSTM model was 0.936. Thus, the LSTM achieved an RMSE reduction of 84.394.

Stock	**RMSE**		**% Reduction**
	ARIMA	**LSTM**	**in RMSE**
N225	766.45	105.315	-86.259
IXIC	135.607	22.211	-83.621
HSI	1,306.954	141.686	-89.159
GSPC	55.3	7.814	-85.869
DJI-Monthly	516.979	77.643	84.981
DJI-Weekly	287.6	30.61	-89.356
Average	511.481	64.213	-87.445
MC	0.81	0.801	-1.111
HO	0.522	0.43	-17.624
ER	1.286	0.251	-80.482
FB	0.478	0.397	-16.945
MS	30.231	3.17	-89.514
TR	2.672	0.569	-78.705
Average	5.999	0.936	-84.394

Figure 5: The RMSEs of ARIMA and LSTM models (Siami-Namini et al., 2018).

The LSTM model clearly outperformed the ARIMA model when implemented by Siami-Namini et al. (2018) on financial and economic-related time series data. Siami-Namini et al. (2018) advocates for the "benefits of applying deep learning-based algorithms and techniques to the economics and financial data," and they hope to further investigate the improvement achieved by deep learning algorithms when compared to more traditional modeling methods.

5. Other LSTM Applications to Time Series Data

In addition to the application of a Rolling LSTM model to the economic and financial datasets by Siami-Namini et al. (2018), there are many other successful applications of LSTM-based models to time series forecasting, both in scholarly journals as well as online articles.

For example, Michael Keith provides a simple exploration of an LSTM model using airline data with the Python package tensorflow (Keith). He was able to successfully predict the number of passengers per month using an LSTM-based model. More specifically, the model achieved a training loss of less than 0.1 and a validation loss of less than 0.05. After hyperparameter tuning and model selection, Keith's model outputted predictions where all but two of the points fell outside of the model's 95% confidence interval.

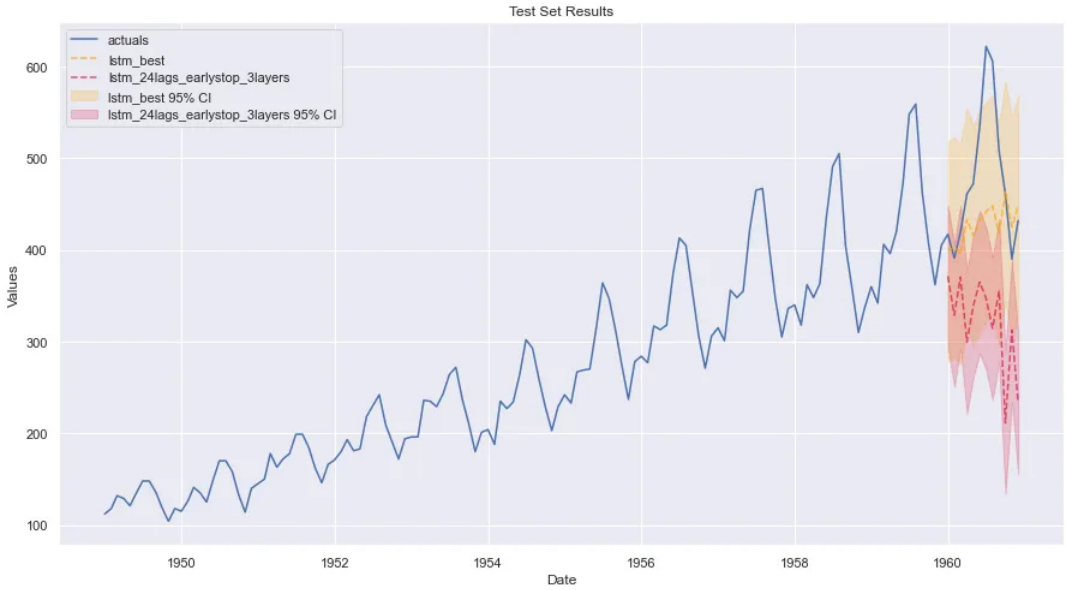

Figure 6: LSTM Model Results on the Testing Set (Keith).

6. Conclusion

This paper provides a theoretical introduction to ARIMA and LSTM modeling techniques for time series forecasting. The methodology and findings from *A Comparison of ARIMA and LSTM in Forecasting Time Series* by Siami-Namini et al. (2018) are summarized. Finally, a brief synopsis of another successful application of an LSTM-based model to time series data is given.

References

Taha Binhuraib. An introduction to time series analysis with arima. Towards Data Science.

Rian Dolphin. Lstm networks — a detailed explanation. Towards Data Science.

Michael Keith. Exploring the lstm neural network model for time series. Towards Data Science.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. *17th IEEE International Conference on Machine Learning and Applications*, 2018.

Tony Yiu. Understanding arima (time series modeling). Towards Data Science.