Time Series Forecasting with Deep Learning Models

Ryan Silva, Eric Steen, Orion Darley – Stanford University

May 2019

Abstract

Deep Learning models are increasingly used for a variety of time series forecasting applications and show significant promise as the industry-leading methodology for the foreseeable future. In particular, Deep Recurrent Neural Networks (RNN) with Long Short Term Memory (LSTM) are the most widely accepted Deep Learning approach for time series. In this paper we approach the problem of predicting Bitcoin (BTC) prices with RNNs and evaluate their efficacy compared to widely accepted traditional time series approaches to modeling and forecasting using Autoregressive Integrated Moving Averages (ARIMA) and Vector Autoregression (VAR) models. We show that not only is Deep Learning an effective time series forecasting tool, but the approach outperforms the ARIMA and VAR models by a wide margin.

1 Introduction

1.1 Overview

Financial time series are especially difficult problems to model and usually require in-depth statistical analysis and assumptions on stationarity and environmental variables. Traditional modeling approaches such as ARIMA and VAR are generally accepted to be able to capture short term trends in the market under the set of assumptions they impose on the underlying time series, however modeling long term trends in the market is still a largely unsolved problem.

We apply deep learning to bitcoin price and technical analysis indicators and compare and contrast with these traditional modeling techniques. In particular, we explore the ability of LSTMs to predict the future of a financial time series based on its history, and how well these models are able to generalize to unseen portions of the time series. We choose LSTMs because of their ability to model sequence data over long horizons. For this problem, we choose input sequence lengths that are significantly long (60-130 time steps) to harness the modeling power of LSTMs.

Financial time series prediction is an interesting sector to apply deep learning models to because 1) the exploration space of hyper-parameters for this particular problem is vast, and well tuned deep learning models could have the ability to significantly outperform today's traditional approach to financial modeling, and 2) accurate models of the stock market can generate new understandings into the influences of the stock market, and generate significant economic impact.

1.2 Traditional Time Series Modeling

The purpose of this paper is not to present the full-length aforementioned time series analysis, but to introduce the fundamental concepts of deep learning which also the fundamental research tool described here. The utility of classic time series models, such as ARIMA and VAR, requires stringent assumptions regarding distribution testing (Shaipro-Wilk tests), time series stationarity and decomposition techniques, and autocorrelation testing.

For this research paper we used the traditional time series modeling approach recommended by Brockwell and Davis [4]. Economic and financial time series fields are known for their complexity and intemperate residuals, it is essential to know the elements and proper statistical testing of the time series prior to the utility of classic time series modeling. Because the main focus of this research is on the application of LSTMs for a currency market, previous examination of time series is necessary.

2 Related Work

Bitcoin prediction introduced in this research paper follows similar modeling approaches proposed by Andrew [1] who compares machine learning modeling and LSTMs to predict various financial markets primarily on Open, High, Low, Close (OHLC) data. Andrew suggests framing predictive models using the mid and close prices, however this paper only explores predicting the close price.

Akita, Matsubara, Uehara, and Yoshihara [2] propose a combined stock forecasting application that converts newspaper articles into distributed representations with "Paragraph Vectors" and combines these results with predicted stock values, economic indices, company price-earnings rations, and other data inputs from LTSMs to predict future price movements.

Zhuge, Xu, and Zhang [12] also use a unique multi-stage modeling approach, proposing a robust multi-stage LSTM model that integrates blog sentiment analysis, quantifiable public opinion, and stock and macroeconomic predictors to predict certain financial

markets.

Stationarity & White Noise

White noise is an important element within time series forecasting. Sequences of random numbers is the essence of white noise within time series and if the time series is truly white noise, it cannot be forecasted. If the forecast errors are not a series of random numbers, tests such as the Box-Ljung test can suggest improvements which can be made to the forecast model. Applications in financial and economic forecasting suggests that real forecasting problems and solutions account for these sequences of random numbers.

Crone [6] examines white noise in time series forecasting with Forward Propagation Neural Networks by observing distributions of symmetric and asymmetric error functions which provided insight into minimizing relevant white noise patterns and providing meaning to what appeared to be white noise in traditional time series lenses.

Cost Functions

Chong, Han, and Park [5] showed that an MSE cost function together with an LSTM was successful in predicting real-valued returns in the Korean stock market. In their study, MSE was used to model stock price movement as a highly nonlinear function of the history of the stock market, mixed with random Gaussian noise. We explore the viability of using the mean squared error (MSE) between the true BTC stock price and predicted model price as a cost function. By using this cost function, we aim to model and forecast the trend of the BTC stock price.

LSTMs and Stock Prediction

Numerous studies have shown the viability of LSTMs in modeling financial time series. Fischer and Krauss [7] show that LSTMs trained on S&P constituent stock data outperform other machine learning methods such as fully connected networks and logistic regression. Nelson et. al. [10] uses LSTMs trained on technical indicators to predict a binary indicator of positive or negative price movement over a set number of time steps with some successful results. Kim T. and Kim H. [8] use an LSTM-CNN on a combination of price data and candlestick image data to obtain superior results to just using price data or image separately, indicating that feature representation can help the quality of LSTMs model financial time series.

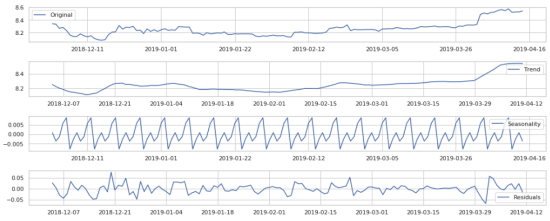
3 Deep Learning Methodology and Approach

Long short-term memory is one of several versions of Recurrent Neural Network (RNN) architecture. In this section, we introduce LSTMs for forecasting the btc closing price, because LSTMs do not require previous information of a time series structure – due mainly to its black-box properties [11]. LSTMs are less subject to time series stationarity. Popular research suggest that these models are more flexible when working with non-stationarity for forecasting [3]. Furthermore, the approach of modeling data with traditional time series models can be simplified by simply importing the data directly into the model with limited amount of EDA, data preparation, and traditional time series analytical knowledge.

The following steps were taken to measure the predicted values distance from ground truth:

1. Imported bitcoin pricing from 2012-current, to include open price, close price, high and low prices, volume transacted, and percentage change over a daily interval.

2. Plotted the distributions of each and determined the threshold for the training data. Most of the data is heavily skewed as to where past price changes do not necessarily represent the current trend nor the near-future trend of Bitcoin prices. This includes testing features and time lengths for stationarity, autocorrelation and partial autocorrelations, seasonal and trend decomposition, and white noise.



3. Standardized the data and created the training and test structure.

4. Built several models using various parameters (various activation functions, hidden nodes, dropout rates, batch sizes, etc.).

5. Benchmarked the predicted values for four models versus the actuals.

3.1 Dataset

For the Bitcoin index, three categories of daily variables are used as model predictors. The first category contains historical Bitcoin trading data, such as the open, close, high, and low bitcoin prices of the day or hour. The second category consists of volume and percentage change in price recorded during trading. These variables are widely used in stock and commodities trading. We also examine feature engineered variables as the third type of inputs. Feature engineered variables include attributes' variation, technical indicators, exponentially weighted smoothing averages, seasonal decomposition, and trend decomposition, etc.

3.1.1 Dataset Split

Both traditional and deep learning models use the same training and test or holdout windows. The training window is from January 1st, 2017 to February 2nd, 2019 and the test or holdout window is from February 3nd, 2019 to March 10th, 2019. The training window could have been longer, however it was excluded before January 1st 2017 due to lack of information and contributed to the model's inability to distinguish white noise from meaningful information.

Each dataset contains the features Close, High, Low, Open, Volume. The final week or our efforts included significant training on data with volume removed, as recent research suggesting a high amount of variance in reported volume data across exchanges [9], reminding us of the huge economic opportunity for data providers that emphasize transparency and integrity.

We notice early on that changing the size of the test set brought fluctuating levels of loss. One interesting finding is that across a range of test set percentage of split between 5% and 50%, we found that our model results were very sensitive to the size of the test set with the best performing split at around 72% train and 28% test.

3.2 Feature Engineering

We explore a variety of data pre-processing and augmentation techniques gathered from the literature, and implement them as part of our EDA (Exploratory Data Analysis). We explored technical indicators, which are often used in the financial sector to make trading decisions and offer insight into the movement of the time series. We experimented with the Relative Strength Index, and Bollinger Bands. We also computed a seasonal, trend, and residual decomposition, as well as a 30, 50 and 200 day moving average. However, our feature engineering resulted in preferring closing Price and High Price for simplicity to ensure model sensitivity to the most widely distributed data features in the financial asset space.

3.3 Architecture

We model this problem as a many-to-many real-valued prediction task. One well performing architecture we use is an encoder-decoder LSTM with a 1D Convolutional layer to begin with. The Convolution layer has a filter size of 10 time steps, and relu activation, with no pooling. The output is passed to an LSTM encoder layer, and produces an encoded version of the data, having a variable size based on our hyper parameter tuning. The output of the encoder is then replicated and fed to the decoder as the input for each time step. The decoder makes a single prediction at each time step, where the output at each time step is passed to a two layer fully connected network with a single output unit with linear activation. The cost function used is Mean Squared Error (MSE) between the true price time series and the predicted time series, over the number of predictions forecasted by the model. All models were trained with the Adam optimizer.

3.4 Hyper-parameter Tuning

The hyper-parameter decisions that were most effective at tuning the model were smaller number of hidden units per layer, dropout, learning rate, and batch normalization. We explored many more hyper-parameters including number of layers and activation functions, and learning rate decay early on but once an optimal tuning crystallized we had a stable base to work off of. We first notice that more than 300 epochs was not generally providing further decreases in loss, and 150 was more than enough epochs across most hyper-parameter combinations.

We methodically established ranges for candidate hyper-parameters that were showing most impact on loss and mean absolute percentage error in results early on in testing(these include dropout, learning rate, hidden units, number of layers). Once the best among the candidates was confirmed, we enumerated all potential combinations to fine tune the remaining parameters within nested loops (combinatorial search). The method is similar to the following pseudo-code, but starting with many more hyper-parameter values, especially in the outer loop, which were eventually narrowed down:

```
for btch, units in [(w, 13, 15, 35)]:
    for dpt in [0.1, 0.2, 0.3]:
        for lr in [.00001, .001, .01]:
            PriceRNN(
                wlen=wlen,
                flen=flen,
                dropout=dpt,
                epochs=150,
                batch_size=btch,
                units=[units] * 4,
                testpct=0.28,
                lr=lr,
                datadir="/data",
            ).run()
```

Well over 200 hrs of training narrowed down the most promising hyper-parameter values as detailed in the following sub-sections.

3.4.1 Learning Rate

Many learning rates were tried in the range .00001 - .1, with .001 beating out every other order of magnitude.

3.4.2 Dropout

Dropout of 0.4 improved the volatility of the model significantly from baseline (sans any regularization), but after tuning, dropout was most effective at between .1 to .2

3.4.3 Hidden Units

We found a lower number of units was most effective for increasing accuracy, reducing loss, and increasing overall stability of our metrics. 20-35 hidden units were the best performing.

3.4.4 Batch Size

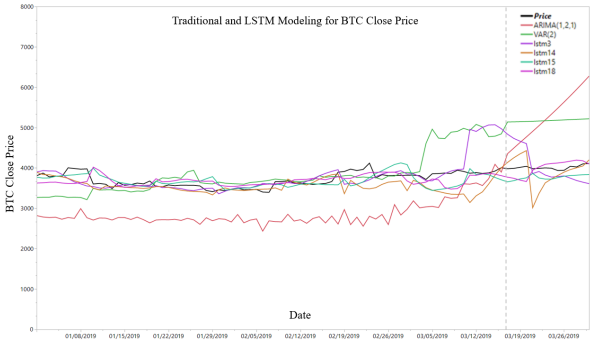
A batch size of between 5 and 40 worked best.

3.5 Results

We evaluated our top 25 models with mean absolute percentage error (MAPE), which is defined as:

$$\frac{1}{n}\sum \frac{|Actual - Forecasted|}{|Actual|}$$

The below chart shows our top models and their predictions vs. the traditional time series models of ARIMA and VAR, along with the actual price of BTC.



As can be seen above, LSTM no. 14 outperformed the top 25 models with a MAPE of 0.022 on the training set and 0.005 on the test dataset. Fourteen other LSTMs outperformed the top performing VAR model and the four ARIMA models ranked at the bottom of the top 25 models list.

Model Name	Features	# Hidden Nodes/layer	Dropout	Sequence Length	Training Score	Test Score
lstm14	13	50	38%	100	0.022	0.005
lstm18	13	300	42%	130	0.003	0.042
lstm15	13	250	41%	120	0.067	0.051
lstm3	13	50	10%	100	0.122	0.083
VAR(2)	4				0.269	0.293
ARIMA(1,2,1)	1				0.527	0.477

In general, we notice that all LSTM models we trained in our hyper-parameter search are able to model the trend of the testing period early on to some extent. However the models with inferior hyper-parameters selections tend to lose the trend as the number of time steps in the prediction increases. More interestingly, these inferior models lose the trend at about the same time that the traditional models also start to drift away from the trend.

The models we found to have good sets of hyper-parameters were able to continue modeling the trend quite well all the way through the testing period, and this was the main difference between our top models and the rest of the models we compare them to.

4 Conclusions

Deep Recurrent Neural Networks can be highly effective in tactical trading decision making support. There are several main advantages to using deep learning models over traditional time series to include: LSTMs have the capability of storing time lags and errors in the signal, which can be stored for long periods of time. LSTMs can also store and use specific historical events, whereas traditional models perform well with more recent events and lacking any memory capabilities at all.

These models also outperformed traditional models with raw data ingestion and did not require a methodology to diagnose traditional time series phenomenon. The disadvantages of using deep learning models for time series as opposed to traditional methods include:

1. Deep learning models can take longer to run. Our LSTMs required exponentially more data than the traditional methods with regards to obtaining out-performing and accurate models.

2. Deep learning models have several hyper-parameters to tune where as traditional models have few parameters to tune, but can have longer EDA and data preparation phases.

Moreover, both traditional and deep learning models were unable to forecast accurately longer than 30 days into the future. For this study, the ARIMA and VAR models could not accurately reproduce and forecast 14-day windows well with this complex data due to large amounts of white noise, nonlinear trends, and complex data. With regards to the LSTM model and their nonlinear activation functions, these models fit the data best. The Best-in-Class (BIC) model, modeled the non-linear data on the test set with a MAPE of 0.005.

5 Future Work

We are optimistic that further questioning and gathering additional insights along with better feature engineering, more high quality data, rigorous optimization, and additional techniques such as attention models and other emerging research findings will give significant lift to this research in the future.

At a technical level, we hope to expand upon and improve our approach with better architectures and rigorous hyper-parameter optimization (stochastic grid or bayesian search). Our techniques can also be enhanced by continuous improvement in our software engineering practice, including unit and integration testing, and the integration of the "daily sprint" practice. Finally, an increase in automation is desired to avoid large amounts of manual tuning for new similar datasets.

6 Code

Our code is on github at https://github.com/ericsteen/crypto_data

References

[1] Andrew. (2017). A. Mann. *A New Methodology to Exploit Predictive Power in (Open, High, Low, Close) Data. 978-3-319-68611-0.*

[2] Yoshihara A. Matsubara T. Uehara K. (2016). Akita, R. Deep learning for stock prediction using numerical and textual information. 2016 ieee/acis 15th international conference on computer and information science (icis).

[3] Izhak Shafran Andrew Senior, Haşim Sak. Context dependent phone models for lstm rnn acoustic modelling. (n.d.).

[4] Davis R. A. (2016). Brockwell, P. J. Introduction to time series and forecasting. cham: Springer international publishing switzerland.

[5] Han C. Park F. Chong, E. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. expert systems with applications.

[6] S. (n.d.). Crone. Prediction of white noise time series using artificial neural networks and asymmetric cost functions. proceedings of the international joint conference on neural networks, 2003. doi:10.1109/ijcnn.2003.1223950.

[7] Krauss C. (2018) Fischer, T. Deep learning with long short-term memory networks for financial market predictions.

[8] Kim H. (2019) Kim, T. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data.

[9] Hong Kim Matthew Hougan and Micah Lerner. Economic and non-economic trading in bitcoin: Exploring the real spot market for the world's first digital commodity.

[10] (2017) Nelson et. al. Stock market's price movement prediction with lstm neural networks.

[11] Marinaro M. (2002). Tagliaferri, R. *Neural nets: Proceedings of the 12th Italian Workshop on Neural Nets, Vietri sul Mare, Salerno, Italy, 17-19 May 2001. London: Springer.*

[12] Xu L. Zhang G. (2017). Zhuge, Q. Lstm neural network with emotional analysis for prediction of stock price.