Stock Price Volatility Prediction with Long Short-Term Memory Neural Networks

Jason C. Sullivan
Department of Computer Science
Stanford University
Stanford, CA
jsull@stanford.edu

***Abstract*— Stock price prediction in the financial markets is one of the most interesting open problems drawing new Computer Science graduates. A successful price prediction model can be highly profitable in trading equities in the public markets. A similarly interesting and profitable problem is that of volatility prediction. In predicting the volatility of a given stock, a trader can make bets or provide liquidity in the options markets. In this study, we employ a variation of a type of Recurrent Neural Network called Long-Short Term Memory (LSTM) in order to predict stock price volatility in the US equity market.**

Keywords—Deep Reinforcement Learning, Trading, Volatility

I. Introduction

Stock price time series possess some unique and frustrating characteristics that makes them particularly difficult to analyze. They are nonstationary (regimes change over time and across seasons), autocorrelated (the phenomenon of "momentum" trades), and subject to imperfectly distributed information across market participants, among other things.

Financial time series are notoriously noisy, due in part to the complexity of the interrelationships between different financial assets, economic developments, and participants in the market. A significant move in a stock could just as easily be due to a large pension fund taking a position as it could be due to some merger announcement or regulatory change.

Adding even more complexity to the problem of portfolio management is the dynamic of stock volatility—that is, the degree of variability of a stock's price over some finite period. Predicting volatility is important for many reasons. An accurate picture of a portfolio's volatility is necessary for an investment manager to manage portfolio risk and meet investment criteria according to the fund's investment mandate.

It is likewise important for options traders and financial product structurers, whose financial instruments are priced in part on expectations of volatility. There are many traders whose entire strategy relies on their ability to predict a more accurate picture of volatility than the rest of the market, thereby allowing them to exploit a form of arbitrage.

As such, in this study, we aim to extend existing work on the use of neural networks to predict stock price volatility. The goal of this work is to advance the understanding of novel methods of volatility prediction so as to positively impact the stability of the overall financial markets—with participants better informed about their portfolio risk, those participants can do a better job of managing risk and avoid contributing to future stock market meltdowns.

II. Related Work

The traditional econometric approach to volatility prediction of financial time series is known as a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model [1]. One of the reasons this model is so popular is that it accounts for heteroskedasticity of financial time series—in other words, the variance of error terms varies as a function of some process, which makes regression modeling very difficult. It also accounts for autocorrelation in financial time series.

In recent years, and at an increasing pace, market professionals are turning to Machine Learning and Deep Learning to gain insight into market dynamics, and in some cases, even create automated trading systems based on these algorithms. While GARCH remains the gold standard for volatility prediction within old-line financial institutions, new Deep Learning methods such as LSTM are becoming increasingly popular approaches to this problem.

Long Short-Term Memory (LSTM) is a variant of a recurrent neural network commonly used for time series predictions, first introduced in 1997 as a method of dealing with the vanishing gradient problem encountered in traditional RNNs [2]. LSTMS are often used for certain problems in Natural Language Processing, since a series of sentences could be viewed as a time series with some degree of autocorrelation since words are often dependent on the words before and after them.

The key idea in this experiment is that LSTMs are uniquely suited to handle time series with varying time periods between "significant" events because they are architected to maintain a "memory" of older data points in the time series. This is highly useful in financial time series analysis because significant stock price moves can be rare and sporadic, and also because the autocorrelated nature of financial time series makes having a cell "memory" useful in training and testing the network.

To this end, an LSTM unit contains a memory cell, along with input and output gates to control the flow of information into and out of the LSTM unit. One modern incarnation of the LSTM also includes a "forget gate" [3] which allows an LSTM unit to reset its state so as to not grow continually. Another variant, called the Gated Recurrent Unit (GRU) lacks an output gate [4].

As discussed above, LSTM's ability to handle time series with inconsistent timing between meaningful events makes it an ideal candidate for time series prediction for financial assets. Indeed, work has been done using LSTM to predict financial time series in the stock market, using both traditional time series analysis inputs as features and using technical analysis metrics as features [9]. The traditional LSTM has also been used successfully to address the problem of volatility prediction in the stock market [5][6], as have traditional RNNs [7].

LSTMs have been shown to outperform traditional GARCH models in this context. We aim to extend this work by using an alternative set of engineered features and by altering the architecture of the neural network, benchmarked against a simple LSTM network, to better suit the nature of financial time series data.

III. Dataset and Features

Our primary data source is from Kaggle (labeled "Huge Stock Market Dataset") and provides over 18 years of daily Open, High, Low, Close, Volume, and Open Interest data for individual US stocks and ETFs [8]. The full dataset is 826MB, consisting of text files representing time series for over 8300 entities, including over 7000 single stocks and over 1300 ETFs.

Lengths of time series vary by instrument based on when that instrument started trading in the market, and whether it still trades (or was acquired, delisted, etc.). We make our training data among different stocks as comparable as possible by focusing only on stocks that were traded between 2013 and 2018.

Daily data for this experiment is preferable to other frequencies, as intraday (tick-level) data is typically far too noisy with too little predictive value, and data from longer intervals is less practically useful. Many new features such as momentum indicators and candlesticks (both of which are commonly used in stock chart analysis) can be engineered from daily data.

In this study, we focus on a universe of the 1000 largest and most liquid single stocks available in the dataset. For each stock, we engineer the same feature set and target variable, training our neural network on all stocks in the prescribed universe so as to generalize our model, rather than just focusing on learning one stock.

Our features consist of several transformations of the given Open, High, Low, Close, Volume data. We use Log Return, Log Volume Change, Log Trading Range (high vs. low for a given trading day), Previous 30-day Volatility, Previous 10-Day Volatility, and GARCH forward-looking 10-day volatility prediction as our features.

Financial time series data requires pre-processing in order to address some nuances that would otherwise make predictive analytics impossible. For instance, financial time series are nonstationary—in other words, their distributions can change over time, often as a result of general market dynamics ("bull" vs. "bear" market) or fundamental changes to the underlying company (increasing debt levels, changes in management, supply chain consolidation).

The common method of addressing this non-stationarity is to not use raw price series as inputs, but rather the series of price *changes* or percentage returns. Both methods are a form of price differencing, which create stationarity in a non-stationary time series. We do the same with daily trading volume. We also normalize the data by taking the logarithm each element of the time series, so as to scale the data.

For the GARCH model, we train a new model for each separate stock. For each of those stocks, we fit a new GARCH model at each successive time step using all previous day, walking forward, as we would do if using the GARCH model on its own to trade. We train the GARCH models with parameters p=15 and q=15, with a horizon of 10 trading days (Figure 1).

For our target variable, we use the average volatility of the 10 days following a given data point. The motivation is that predicting a 1-day volatility doesn't make much of a difference and will be too noisy to train effectively, and predicting a 30-day volatility is not nearly as useful with a shorter-term trading horizon as part of one's trading strategy (Figure 2).

Given the relatively small size of the data, due to the fact that each data point represents a daily closing price, we need to tailor our train/test split to this data scarcity. We are using daily price data from a 5-year period, and despite having 1000 stocks in the chosen dataset, this is still not a great deal of data. As a result, We choose to go against recent wisdom in the Deep Learning community and stick to the 80/20 train/test split.

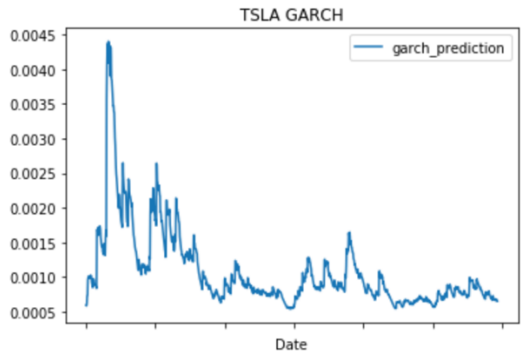

Figure 1: TLSA GARCH model results

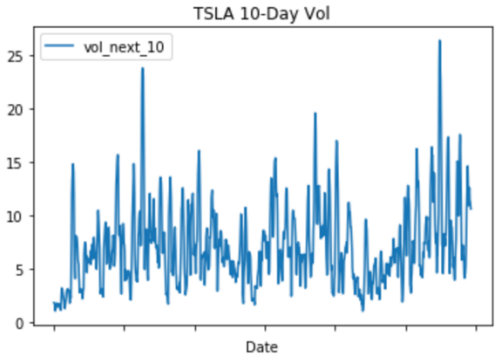

Figure 2: TSLA 10-Day Forward Volatility

IV. Methods

Because we are dealing with time series data, we cannot simply randomly shuffle datapoints into train/test splits. As such, we make our splits first by stock ticker, and then by chronology We will consider data in rolling 30 trading day blocks, closing price volatility levels over the following 10 trading days as our y labels.

Our baseline model (Figure 3) consists of a single LSTM layer with 5 hidden units, followed by a 1-unit output layer. We train this neural network with a batch size of 30 data points. Our chosen loss metric is Mean Squared Error, but we also track Mean Absolute Error and Mean Absolute Percentage Error. We use the Adam optimizer.

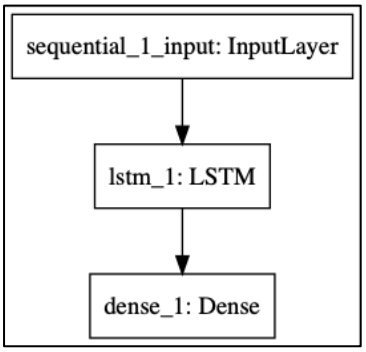

Figure 3: Neural network architecture for baseline model.

Our extended model (Figure 4) is depicted below. It consists of a 20-unit LSTM layer, a 10-unit Dense layer with ReLu activation, and a 10-unit GRU layer. We also include 2 separate 10% Dropout layers.

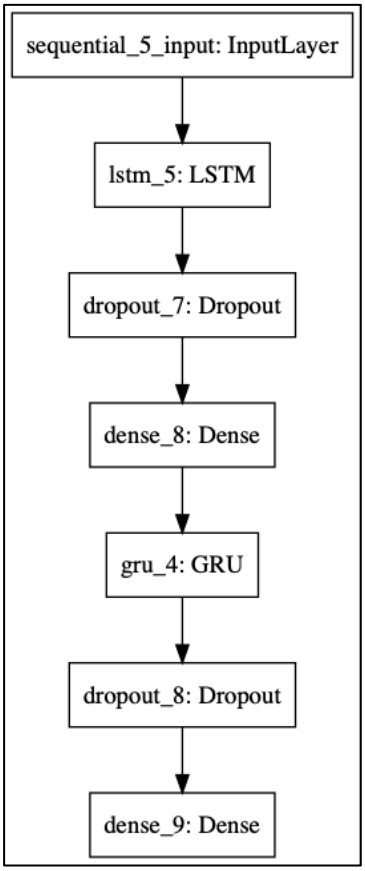

Figure 4: Neural network architecture for extended model

As for implementation in code, employ the Keras framework in Python, as well as Pandas, Numpy and Matplotlib.

V. Experiment/Results/Discussion

Below, we compare the results of both the baseline and extended models. As we can see, the extended network outperforms during training, but the simpler network outperforms during testing (Figure 5). Notably, the extended network also has a greater degree of performance degradation in MSE and MAE in the test set relative to the training set, suggesting a greater degree of overfitting within the extended network. This suggests that a greater deal of tuning is required on the deeper network, and in particular, the increased use of dropout layers could help reduce the variance problem.

Training Set	MSE	MAE	MAPE
Baseline	4172.61	2.339	1975190.75
Extended	3883.5	2.314	8526497.0

Test Set	MSE	MAE	MAPE
Baseline	5167.51	2.459	8722592.0
Extended	4859.47	2.471	37653732.0

Figure 5: Train and test set results.

VI. Conclusions and Further Research

As it turns out, the simpler model actually performed better on this data. It is likely that an even deeper model trained on larger data would have fared better, but in this case, simpler was better.

A great deal of further research can be done in this relatively nascent area of financial time series prediction. There are multiple axes on which we could vary our experimental setup.

The first axis to consider is time series frequency. Our experiment uses daily stock price data as our key input. However, different market participants focus on different frequencies of data. High-frequency trading firms tend to focus on tick-level or order book-level data, creating orders in fractions of seconds. The type of exchange and order book arbitrage that HFT firms focus on relies on high-frequency, very noisy data. The challenges posed by this level of noise are very different, and likely require different data pre-processing and neural network architecture.

Another axis to consider is the asset class of the financial instrument in question. Here, we have focused on "vanilla" US equities. However, there is a wide world of other asset classes that all trade differently. Corporate bonds, for instance, trade at a lower frequency and in larger trade sizes than equities do, with different variables affecting price and yield. Likewise, currencies and cryptocurrencies all have their own nuances, and a neural network trained on one asset class is unlikely to generalize to another.

Another area to consider for future research is alternative and more expansive feature engineering. In our case, we are using transformed versions of the most basic stock data available—price and volume. However, market practitioners and researchers alike do not typically rely only on price and volume data. One area of interest that dovetails nicely with RNNs is trading on news sentiment. Using RNNs to quickly parse and classify news headlines or earnings reports as positive or negative is a fast and effective way to trade on news.

One more possible variation in our experimental framework for future research would be the framing of the target variable. In our case, we frame the problem as a regression problem, with forward-looking 10-day volatility as our target variable. However, this problem could also be framed as a classification problem. If we consider an increase or decrease in volatility beyond some specified threshold (i.e. a 3-sigma move) to be associated with a positive or negative label, with a 0 label representing no significant volatility change, it is possible that this new framework could be more seamlessly integrated into an actual automated trading system, making buy or sell decisions based on predicted labels.

Lastly, it would be worth considering alternative neural network architectures for this problem. More work could be done in tuning our current architecture, in the form of tuning hyperparameters, number of epochs, batch sizes, memory length, and the inclusion/exclusion of "forget gates". Adding or removing LSTM/GRU layers, tuning dropout layers, and experimenting with different activation functions are all worth pursuing.

Acknowledgments

Many thanks to the teaching staff of Stanford University's CS230 Deep Learning course, in particular Advay Pal, for their input in the development of this project.

References

[1] Bollerslev, Tim, 1986, Generalized autoregressive conditional heteroskedasticity, Journal of Econometrics 31, 307–327.

[2] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[3] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: continual prediction with LSTM," *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, Edinburgh, UK, 1999, pp. 850-855 vol.2.

[4] Cho, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. Conference on Empirical Methods in Natural Language Processing* 1724–1734 (2014).

[5] M. Sardelicha and S. Manandhar, "Multimodal deep learningfor short-term stock volatility prediction," https://arxiv.org/abs/1812.10479.

[6] Xiong Ruoxuan, Eric P. Nichols, Yuan Shen, Deep Learning Stock Volatility with Google Domestic Trends, 2015.

[7] Liu, Yifan & Qin, Zengchang & Li, Pengyu & Wan, Tao. (2017). Stock Volatility Prediction Using Recurrent Neural Networks with Sentiment Analysis. 192-201. 10.1007/978-3-319-60042-0_22.

[8] https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

[9] Sang, C., & Di Pierro, M. (2018). Improving trading technical analysis with tensorflow long short-term memory (lstm) neural network. The Journal of Finance and Data Science.