Using Deep Learning to Time the Stock Market

SI 671: Data Mining

Chloe Hull

Project Motivation

- Rise of online trading platforms like Robinhood and historically low interest rates has resulted in increased exuberance for the highly speculative options market
- Traditional finance theory states that arbitrage opportunities should not exist systematically; however, the market is imperfect and such arbitrage opportunities do arise occasionally

- I aim to apply data mining techniques to identify and capitalize on these arbitrage opportunities
- Extensive prior work on algorithmic trading has been done at the fund level, but little attention has been given to individual portfolio analysis; that is where this project focuses

Data

- United States Federal Reserve Economic Data (FRED) was relied upon to conduct the analysis
- Four distinct data sets were used to build features: NASDAQ Index, NASDAQ Volatility, Fed Interest Rate, and Bitcoin Index
- Time-series data incorporated daily metrics for each feature, from 2001 until 2020

Acknowledgement

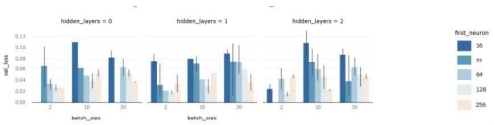
- Thank you to Professor Dhillion & the rest of the SI 671 teaching staff for your support and guidance throughout the semester
- Thank you to the Talos library creators, whose incredible library saved an immense amount of time on model tuning

Objectives

- Utilize machine learning and deep learning methods best suited to make time-series predictions
- Identify market arbitrage opportunities by predicting when dips in the NASDAQ Index will occur:
  - *Can I use machine learning, deep learning, or other data mining techniques to identify the best entry point for investing in the NASDAQ stock index?*
- Use research findings to inform my personal investment strategy

Methods & Evaluation

- Data pre-processing to impute missing values, concatenate data frames to generate new features, and drop dates with erroneous data
- Classical ML models tested initially:
  - Linear Regression, Support Vector Machines, Ensemble Methods
  - Terrible performance, which was expected
- Deep learning utilized for final model (Long Short Term Memory, a specialized type of Recurrent Neural Network):
  - Further pre-processing required to transform data into tensors
  - Conducted extensive hyper-parameter tuning and evaluation of feature importance using the Talos library, tuning hyper-parameters such as batch size, hidden layer nodes, optimizer, and more

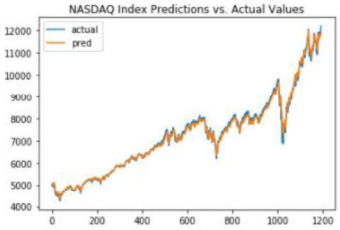


- Talos packaged enabled evaluation of over 100 distinct hyper-parameter pairings, conducting grid search and cross validation to obtain the best hyper-parameters, which were used in the final model
- Evaluation metrics used included R^2 and MAE, as well as visualization of the predicted vs. actual values

Analysis

- Traditional ML models performed terribly, as expected, with negative R^2

- Using three time-steps and after hyper-parameter tuning, I was able to achieve the following performance:
  - R^2 of .994 and MAE of 94.49



- Hyper-parameter tuning allowed performance of LSTM to improve by 15%, from .86 with a baseline LSTM to .99 with the final mode

Conclusions and Future Work

- LSTM allows for very accurate prediction of next day NASDAQ index performance
- Use as a trading tool requires daily data updates, followed by re-training of the model (which takes over 1 hour); as such, while the results are promising, implementation is not realistic at this stage
- Future areas of work include applying this framework to individual stocks, which are significantly more volatile that the index

Using Deep Learning to Time the Stock Market

Chloe Hull
School of Information
University of Michigan
Ann Arbor, MI 48109

1 Introduction

With the rise of online trading tools like Robinhood, historically low interest rates and a perpetually climbing stock market, more individuals are partaking in the highly speculative practice of options trading. In traditional financial and economic theory, we know that arbitrage opportunities should not exist systematically. However, we also know that the market is imperfect and as such these opportunities do arise from time to time. This paper aims to explore the application of data mining techniques to optimize stock trading strategies.

While the rise of day trading amongst individual investors is a relatively new trend, the application of advanced data mining techniques to the financial markets is certainly not new. Hedge funds, investment managers, and advanced researchers have developed state of the art algorithmic trading approaches that have been in production for many years.

This paper does not aim to produce any novel algorithmic trading approaches, but rather to explore how an individual can apply data mining techniques to a small scale, individualized trading portfolio. Many of the existing research efforts have gone towards optimizing for large-scale portfolios and assume the availability of extensive capital; my research will focus only on the applicability to an individual, early stage investors portfolio. Personally, my hope is to use these results to guide my own investment strategies moving forward.

More specifically, the question my research aims to answer is**: Can we use machine learning, deep learning, and other data mining techniques to identify the best entry point for investing in the stock market?** My goal is to predict when short-term dips will occur so that I can use those dips as an entry-point into the market.

2 Experimentation

2.1 Approach

With the goal of predicting the best entry point into the stock market (in other words, can we time the market?), I selected the NASDAQ index as a proxy for the overall market. I first developed models to predict the overall market trend, then planned to focus on stocks of particular interest to see if this approach can be done on a company specific level.

Before diving into modeling, I conducted some exploratory data analysis to get a better sense of the data. I plotted the four indicators I incorporated to see if there appeared to be any correlation between the trend lines; there was, unsurprisingly, a great deal of variation between the plots given their vastly different data points (eg. interest rates only fluctuate between a few percentage points, whereas the NASDAQ index has climbed over 10,000 in the past decade).

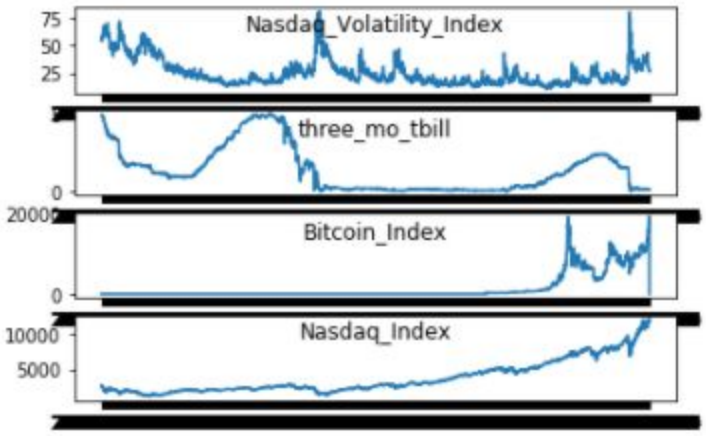


Figure 1: Distribution of Data

I began by evaluating the performance of traditional machine learning models, with the assumption that time-series specific models will perform better given that we are working with time-series data. Unsurprisingly, the basic regression (Linear, Lasso, Ridge, KNN) and ensembling methods (Adaboost, XGboost, Gradient Boosting) performed very poorly. I quickly shifted my focus to deep learning models that are more well suited to temporal data.

Traditional recurrent neural networks struggle to incorporate time-series data with more than a few time lags; the Long Short-Term Model (LSTM) was designed to solve this problem. An LSTM model is a specific type of recurrent neural net that incorporates long term dependencies from time-series data, improving upon vanilla RNNs that do not retain memory for longer term time lags. The LSTM moves through each time step, updating its memory and forgetting gates and using these internal gates to map input sequences to the output layer.

I initially used 3 time-steps, meaning the LSTM is trained on the previous three days of indicators. I tested using anywhere from 1 to 30 time-steps and found that using a shorter number of time steps performed best.

To conduct hyper-parameter tuning, I utilized the Talos package, which is designed to enable hyper-parameter tuning with any model within Keras. This is achieved by building a model then passing in a dictionary of parameter values into the Talos scan, which parses the hyper-parameters and performs grid search. The best performing hyper-parameters are easily extractable, automating much of the traditionally painful hyper-parameter tuning process.

After the model has been constructed and predictions made, I had to reformat the results to make predictions at scale. I did this using the inverse transform of the predictions and then compared the predictions to the actual values to evaluate performance of the model.

2.2 Data Set

The data was taken from the United States Financial Reserve Economic Data (FRED). The data utilized includes the NASDAQ index, NASDAQ volatility index, the 3 month treasury bill index (which I use as a proxy for the current interest rate), and Bitcoin. Each of these data sets includes daily time-series data for the past 15 years.

Data preprocessing included imputed missing values as 0, combining datasets into one data frame, dropping dates with clearly erroneous data, and preparing the data for the LSTM. To convert the data into a tensor for use by the LSTM, I utilized open source code to obtain three time-steps for each row of data, with the last column being the NASDAQ index on that date (the value our LSTM will predict). I then passed the tensor through MinMaxScaler to normalize the data, resulting in a tensor with NASDAQ Index, NASDAQ Volatility, Interest Rate, and Bitcoin Index for the prior three days and the NASDAQ Index for that date.

Due to the time-series nature of the data, and to minimize the risk of data leakage, I used the sklearn module TimeSeriesSplit to obtain the Train and Test sets.

2.3 Evaluation Method

Mean absolute error (MAE) and R-squared (R^2) used as the primary evaluation methods. Originally, MAPE was explored as an alternative, but this may cause the optimization not to converge if using Adam as the optimizer (which was highly likely for this model). As such, MAE and R^2 were the primary metrics used to consider the performance of a model.

MAE is a natural evaluation method because it evaluates the absolute difference between two variables; in this case, between the model's predicted value and the actual value.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Additionally, R^2 is traditionally used as a metric for linear regressions; while this problem is not a linear regression, it does follow a linear trend-line and as such, R^2 is an appropriate measure. R^2 evaluates the goodness of fit of the model.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Finally, because evaluation metrics can at times be misleading, I graphed the predicted and actual values to visualize the trend lines for the better performing models.

3 Analysis

With a goal of surpassing R^2 of .9, I quickly disregarded the traditional ML models (linear regression, ensembling, SVM, etc.). Each of these models performed very poorly, yielding negative R^2 results and very high MAE (around 4,000). A negative R^2 means that the model's predictions are worse than if I were to just guess the mean value to date. While I anticipated these models would not perform well given the time-series nature of the problem, I was surprised at just how poorly they did. I shifted gears very quickly to focus on deep learning methods.

Figure 2: Linear Regression Model Performance

I did not evaluate any deep learning methods other than LSTM, as I know LSTM models are well suited to time-series data and have often been used for stock market research models. As mentioned in the approach section above, 3 time-steps were utilized to build the baseline LSTM model. I used the Keras library's LSTM with 250 hidden layer nodes, a dropout layer of .2 to reduce overfitting, and a dense layer for the output; I used batch size of 15 and 10 epochs to yield the best results for this baseline. Using 3 time-steps, the baseline LSTM resulted in an MAE of 461 and an R^2 of .86. While this was decent performance, in order to be reliable enough to base investment decisions on, I wanted to get closer to an R^2 of .9 or above.

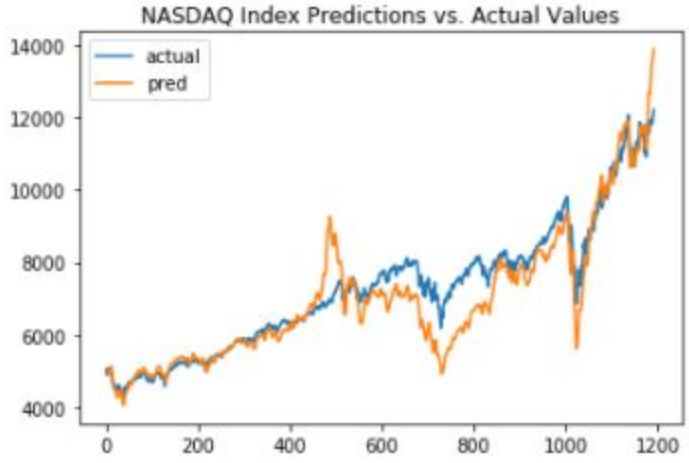


Figure 3: LSTM Baseline Model Performance

Next, I conducted extensive hyper-parameter tuning using the Talos library. During tuning, I evaluated and selected the best hyper-parameters such as learning rate, dropout rate, optimizer, hidden layer nodes, batch size, kernel initializater, loss function, and activation function. The image below shows the performance of various hyper-parameter combinations. Analyzing graphics such as this gave additional context and explainability to the output of the Talos grid search results.

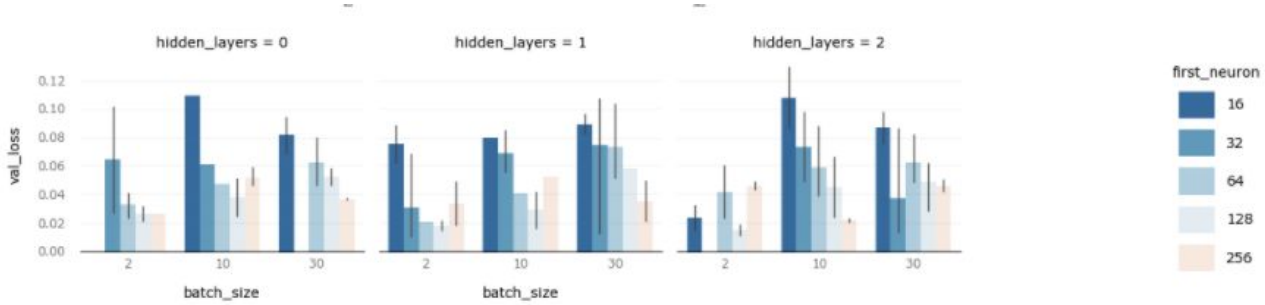


Figure 4: Hyper-Parameter Impact on Performance

After evaluating over 100 different models from a selection of over 25 different hyper-parameters, the final LSTM model was able to achieve an MAE of 94.49 and R^2 of .994, which was far better than the baseline model.

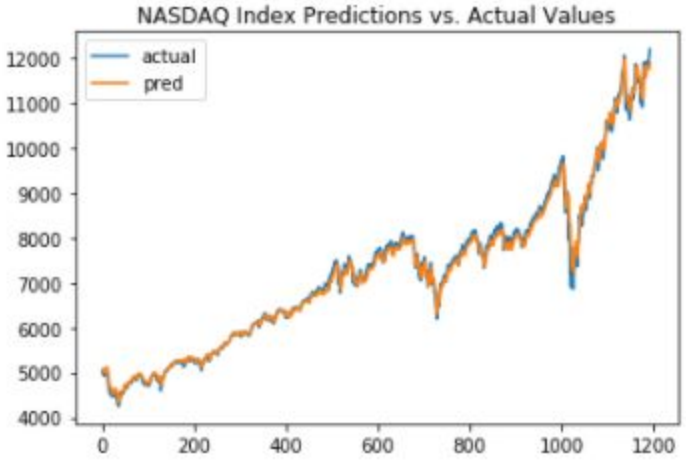


Figure 5: LSTM Performance After Hyper-Parameter Tuning

loss	acc	val_loss	val_acc	activation	batch_size	dropout	epochs	first_neuron	hidden_layers	kernel_initializer	losses	lr	optimizer
.004715	0.00028	0.066075	0.0	elu	10	0.2	10	128	2	normal	<function log_cosh at 0x000002043F933948>	0.26	nadam
.007315	0.00028	0.062374	0.0	relu	30	0.5	10	128	2	normal	<function binary_crossentropy at 0x000002043F9...	0.26	nadam
.003628	0.00028	0.014481	0.0	elu	2	0.0	10	32	1	normal	<function binary_crossentropy at 0x000002043F9...	0.44	adam
.003147	0.00028	0.045520	0.0	elu	10	0.0	10	32	1	normal	<function binary_crossentropy at 0x000002043F9...	0.26	nadam
.007504	0.00028	0.121612	0.0	relu	10	0.2	10	16	2	normal	<function binary_crossentropy at 0x000002043F9...	0.38	adam

Figure 6: Impact of Various Hyper-parameters on Performance

I had planned to incorporate additional features into the model, such as inflation rate, LIBOR rate, unemployment, etc. to further improve performance, but with performance already exceeding my expectations, I decided not to incorporate any additional features for fear that they may introduce noise. As such, I concluded my research for this paper with my final model achieving an R^2 of .994 and an MAE of 94.49.

4 Conclusion

Through the application of LSTM models to the stock market, I can quite accurately model future index performance, even during volatile bull and bear markets. That being said, the LSTM relies upon time-steps from the preceding three days, which limits the usefulness of the model. Actually utilizing this to make investment decisions would require updating our input data and re-running the model on a daily basis to evaluate expected stock market performance for the next day.

Future areas of work include applying this framework to individual stocks, which are typically much more volatile than the index where the reversion to the mean across the entire index counterbalances highly volatile movements on any given day. In order to best model individual stock movements, additional data would likely need to be incorporated. One interesting thought is to incorporate twitter data and perform sentiment analysis to evaluate social media influence on market performance.

Deep learning continues to be applied across the financial industry, with hedge funds dedicating immense resources to the development of novel AI driven trading strategies. As computing resources improve and research efforts continue, I am excited to learn about the novel implementations and their impact on the financial markets and overall economy.

References

1. Brownlee, J. (2020, February 19). A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Retrieved from https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
2. Brownlee, J. (2020, August 27). How to Tune LSTM Hyperparameters with Keras for Time Series Forecasting. Retrieved from https://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting/
3. Chablani, M. (2017, July 15). RNN Training Tips and Tricks:. Retrieved from https://towardsdatascience.com/rnn-training-tips-and-tricks-2bf687e67527
4. Federal Reserve Economic Data: FRED: St. Louis Fed. (n.d.). Retrieved from https://fred.stlouisfed.org/
5. Introduction. (n.d.). Retrieved from https://autonomio.github.io/docs_talos/#introduction
6. Mikko. (2019, May 21). Hyperparameter Optimization with Keras. Retrieved from https://towardsdatascience.com/hyperparameter-optimization-with-keras-b82e6364ca53