Tensor Train Methods for Predicting Financial Market Trends

Matthew Chung [1]

Abstract

In this paper we explore the application of spectral methods in the context of Recurrent Neural Networks (RNNs), for forecasting financial market trends in time series. In particular, we compare Rose Yu's proposed Tensor Train RNN (Yu et al., 2017), with other state of the art architectures used for market forecasting, like Long Short Term Memory Networks (LSTMs).

To motivate our idea, in the real world, we often see that the order in which events occur matters, and that there typically exist some periodic, cyclic trends in what we observe. However, financial markets typically do not present these same characteristics, and we view them as one of the most chaotic/unpredictable naturally occurring datasets in the world.

With stock prices being so heavily influenced by factors like social media, news, politics, etc. Predicting trends proves to be quite a challenge as prices are susceptible to a number of factors outside of just the market environment itself. With this in mind, we propose a novel application of Tensor Train RNNs for learning the nonlinear dynamics in market volatility, and suggest why predicting over volatility, rather than price itself, could lead to some interesting conclusions.

1. Introduction

To reiterate, we consider the scenario in which the data we hope to analyze is given in time series–specifically, stock market data. To explicitly define what "time series" means, we define it as data which indexed by time (i.e. the order in which it is observed matters).

Now, let us more concretely state our dataset and conclusions we hope to make using our model. For our specific case, we will be looking at the past 12 years of historical data from 29 out of the 30 companies tracked in the Dow Jones Industrial Average Index (DIJA) from January 1, 2006 to January 1, 2018. We exclude data for the ticker 'V', or Visa, simply because it does not have the whole 12 years of data. Our dataset was scraped from Google Finance using the python library 'pandas', and came with the open, close, high, low, volume, and name according to their prices in the NYSE (all data measured in USD).

2. Background

2.1. Finance Terminology

In our paper we will be referencing a lot of finance terminology, so rather than defining concepts as they are introduced we will overview the definitions here.

To begin, while engineering our feature set, we chose to look at a few things outside of just the open, close, high, low, and volume. In particular, we saw that the open close, high, and low were highly correlated, and chose to only keep the closing price of each stock, dropping all the others. With the close, we were then able to compute the 7, 30, and 90 Day Moving Averages/Volatilities, as well as the Market Meanness Index.

- **N-day Moving Average:** Average of a stock's closing price over the past n days.
- **N-day Volatility:** Variance of a stock's closing price over the past n days.
- **Mean Reversion:** The theory that asset prices and returns eventually return back to the long-run mean or average of the entire dataset.
- **Autocorrelation:** The mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals.
- **N-day Market Meanness Index (MMI):** An indicator to measure how likely a stock's is to revert to a mean after starting some trend (Lotter, 2015).

In the above definitions, the N-day Moving Average and Volatility are easy to understand, but to explain in a little more detail what the Market Meanness Index does/why we choose to analyze it, we compute it as shown in Algorithm 1.

[1]Department of Computer Science, University of Maryland-College Park, College Park, MD. Correspondence to: Matthew Chung <mchung14@umd.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Consider a price curve with median M. By definition then, half of the observed prices are less than M and half are above. Now if we look at the price between any two intervals of time, say P_{t-1} and P_t, we have 4 cases for what could occur.

1. $P_{t-1} < M$ and $P_{t-1} < P_t$
2. $P_{t-1} < M$ and $P_{t-1} > P_t$
3. $P_{t-1} > M$ and $P_{t-1} < P_t$
4. $P_{t-1} > M$ and $P_{t-1} > P_t$

After looking at these conditions, we see that conditions **1** and **4** represent the price pairs that seem to revert to the median and **2** and **3** represent the price pairs that seem to divert from the median. In our algorithm, we calculate the pairs meeting conditions **2** and **3**, and return the ratio of pairs diverting from the median over the total number of pairs. Therefore, we can think of the MMI, intuitively, as a measure of how likely the market is to be in a phase of mean reversion or "trendiness", where if goes down, it means that over the past n days prices have been moving closer to the n-day moving average more often than they've been moving away.
By using the N-day Market Meanness Index as one of our features then, the idea was that it could potentially make our predictions on future price movement more accurate.

Algorithm 1 Market Meanness Index (MMI)

Input: $data$
$m = data.median()$
$nh = nl = 0$
for i **in** $range(1, len(data))$ **do**
  if $data[i] > m$ and $data[i] > data[i-1]$ **then**
    $nl+ = 1$
  else if $data[i] < m$ and $data[i] < data[i-1]$ **then**
    $nh+ = 1$
  end if
end for
return $100.0 * \frac{nl+nh}{len(data)-1}$

2.2. Nonlinear Dynamics vs. Chaos

In this section, we will be differentiating between nonlinearity and chaos, while also putting our problem into perspective and motivating our reasons for why we believed Rose Yu's proposed Tensor Train RNN (Yu et al., 2017) would be an interesting model to test against financial data.
As we can see in Figure 1, inherent in our data are some increasingly *chaotic* and *nonlinear* dynamics. While previous literature has shown that nonlinear dynamics can generally by modeled using a variety of methods, the trend seen in these papers is that the data itself exhibits some periodic, cyclical behavior.
To define what we mean by all this, we include the following definitions:

- **Nonlinear Dynamics:** Systems governed by equations more complex a linear $y = mx + b$ form.
- **Periodic Trends:** A signal or event that occurs after a specific interval of time.
- **Cyclical Trend:** A similar pattern or trend that continuously occurs over time (not guaranteed to be periodic).
- **Chaos:** Dynamical systems that are highly sensitive to initial conditions.

Some examples of nonlinear periodic, cyclical phenomena are things like climate and traffic. With climate for example, one would expect temperatures to drop during certain times of the year like December of January, and go back up at some point in May/June. But between these times/different seasons, we also notice that we might experience a 50 degree day in December, or some random hail in the summer that seems to follow no trend.
Unfortunately for us, stock prices don't typically follow as obvious periodic, cyclical trends. For example, you might notice that over the past 10 years a company's stock price has seemed to continually increase without any sign of a cyclical or periodic trend.
This introduction of what I like to call "mild" chaos into our dataset, makes our problem somewhat unique. A good example of "mild" chaos in markets is that AAPL's opening price tomorrow could be dramatically affected if a news channel reports that all iPhone XS' have a terrible virus causing them to explode, but there are very few things that could greatly impact tomorrow's weather.
Creating a model that can learn these intricate nonlinear trends in a mildly chaotic dataset, proves to be quite a challenge, and so we look to see if Tensor Train RNNs could solve this problem.

3. Learning Market Trends

In this section, we will be discussing previous work done on learning market trends, as well as overviewing Rose Yu's Tensor Train RNN (TT-RNN), and how her proposed RNN architecture could potentially outperform other existing methods.

3.1. Autoregressive Models

One popular method that a lot of financial models have looked into are Autoregressive Models. In essence, these

Figure 1. Plotted above are the true Close, 30 Day Moving Average, 30 Day Volatility, and Market Meanness Index over the course of 3000 days, for the first 3 tickers (alphabetically) in our dataset.

models make predictions by taking a weighted sum of precious observations to predict the next. We can then write all autoregressive models into something of the form

$$X_t = c + \sum_{i=1}^{L} \rho_i X_{t-i} + \epsilon_t$$

where X_t represents the future we are predicting, $\rho_i ... \rho_p$ represents the parameters of the model, c is some constant, and ϵ_t is some white noise.
Obviously a model of this form is unable to learn nonlinear dynamics, as it models the process X_t linearly.

3.2. Recurrent Neural Networks (RNNs)

With a variety of RNN architectures like DeepAR (Flunkert et al., 2018), MinimallRNN (Chen, 2017) and more, they are all formed on the basic idea that a current state is computed based solely on some number of previous states.
While this does, in a sense, allow the model to "remember" things in the past, these models are restricted in learning more complicated relationships over time. This in turn, makes them less capable of learning higher order dynamics because their transition functions are not able to be as complex.
And that brings us to TT-RNNs, which will hopefully be able to overcome these restrictions that RNNs generally have.

4. Tensor Train Recurrent Neural Networks (TT-RNN)

In their paper "Long-term Forecasting using Tensor-Train RNNs" by Yu et al. (Yu et al., 2017), the authors introduce a novel RNN architecture that rather than looking at just the previous state to compute a current state, looks at the previous L sates and higher order moments, to approximate higher order state transition functions. By doing so, they also prove that it gives their model the ability to make more accurate predictions in nonlinear systems.
To begin, we state the prediction problem as such

$$f(x_0 \dots x_t) \rightarrow (x_{t+1} \dots x_T)$$

Given the previous $x_0 \dots x_t$ observations, can we learn the model f to predict the future $x_{t+1} \dots x_T$ observations?
We begin by looking at the way single cell RNNs compute their hidden states

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}; \theta)$$

$$\mathbf{y}_t = g(\mathbf{h}_t; \theta)$$

Where $\mathbf{h}_t$ is the hidden state at time t and $\mathbf{h}_{t-1}$ at time $t-1$, f is the state transition function, x_t is the input, g is the output function, $\mathbf{y}_t = \mathbf{x}_{t+1}$ is the output, and θ are the model parameters.
Notice that we can also represent the same thing using matrices.

$$\mathbf{h}_t = f(W^{hx}\mathbf{x}_t + W^{hh}\mathbf{h}_{t-1} + \mathbf{b}^h)$$

$$x_{t+1} = W^{xh}\mathbf{h}_t + \mathbf{b}^x$$

Where $\mathbf{b}^h$ and $\mathbf{b}^x$ are biases. f is the state transition function, and W^{hx}, W^{hh}, W^{xh} are transition weight matrices that take something from the input to hidden space, hidden to hidden space, or hidden to input space.
But since we want to look at more than just the previous hidden state when using TT-RNNs, the first thing we must do is restructure the way our *current* hidden state is computed, by generalizing it to a higher order as follows.

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}, \dots, \mathbf{h}_{t-L}; \theta)$$

Second, we must also concatenate out L previous hidden states somehow, so that we can compute the higher order moments between them/approximate the model's state transition function so we do exactly that using what is called an "augmented state" $\mathbf{s}_{t-1}$.

$$\mathbf{s}_{t-1}^\top = [1\mathbf{h}_{t-1}^\top \dots \mathbf{h}_{t-L}^\top]$$

Third, for every hidden dimension, we create a P-dimensional transition weight tensor by modeling any degree P polynomial interaction between hidden states as follows:

$$[\mathbf{h}_t]_\alpha = f(W_\alpha^{hx}\mathbf{x}_t + \sum_{i_1,\dots,i_P} W_{\alpha;i_1 \dots i_p}(\mathbf{s}_{t-1;i_1} \otimes \cdots \otimes \mathbf{s}_{t-1;i_P}))$$

where α indexes the hidden dimensions, i indexes the high order terms, and P is the polynomial order.

Finally, we use the TT-RNN with an LSTM cell, called a "TLSTM" cell, and is defined similarly as:

$$[\mathbf{i}_t, \mathbf{g}_t, \mathbf{f}_t, \mathbf{o}_t]_\alpha = \sigma(W_\alpha^{hx}\mathbf{x}_t + \sum_{i_1,\ldots,i_P} W_{\alpha;i_1\ldots i_p}(\mathbf{s}_{t-1;i_1} \otimes \cdots \otimes \mathbf{s}_{t-1;i_1}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \circ \mathbf{f}_t + \mathbf{i}_t \circ \mathbf{g}_t$$

$$\mathbf{h}_t = \mathbf{c}_t \circ \mathbf{o}_t$$

where $\circ$ represents the Hadamard product.
Note that TT-RNN is a basic unit which can be implemented into any architecture. But for Yu's specific case, she implements her TT-RNN as a module for the Sequence-to-Sequence (Seq2Seq) framework (Sutskever et al., 2014).
It remains then, to show how this method of using Tensors will be tractable when the number of parameters in W_α obviously grows exponentially as $O(HL^P)$, with respect to hidden size H.
Well, by using tensor networks to decompose the large weight tensor into ones of lower rank, we can reduce the number of parameters that must be estimated while also not losing any information. The decomposition itself is performed as follows:

$$W_{i_1\ldots i_P} = \sum_{\alpha_1\ldots\alpha_P} A^1_{\alpha_0 i_1 \alpha_1} A^2_{\alpha_1 i_2 \alpha_2} \ldots A^P_{\alpha_{P-1} i_P \alpha_P}$$

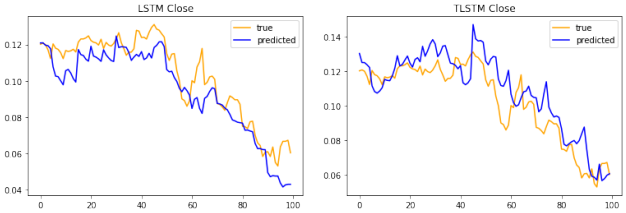
with $\alpha_0 = \alpha_P = 1$, and $\{A^p \in \mathbb{R}^{r_{p-1} \times n_p \times r_p}\}$.
Yu also proves that by specifically using what's called a Linear Tensor Network (LTN) or "Tensor Train", decomposing the weight tensor becomes tractable and does not suffer from the curse of dimensionality (Yu et al., 2017).

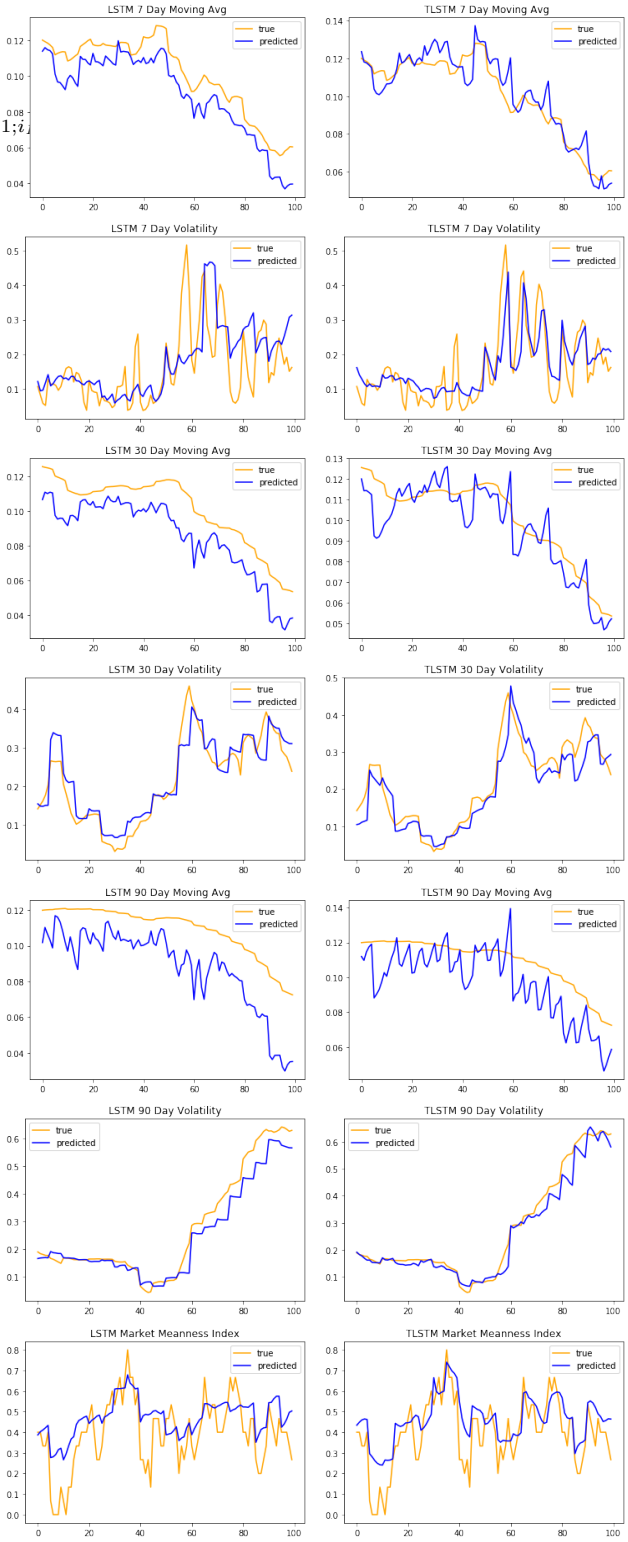
5. Results

For the bulk of this section, we will be comparing our final results between LSTMs and our new TLSTMs. We choose to use LSTMs as the baseline for comparing the TLSTMs against, simply because in recent literature the most successful models used them as their base module.

5.1. Prediction Results

Note that for all predictions models used the past 50 days to predicted the next 100, and that predicted outcomes are all 0-1 scaled.





Overall, we saw that the TLSTM model did in fact outperform the LSTM model most of the time. After computing the Mean Squared Error (MSE) of both models over

on 20 trials, the average testing loss for the TLSTM was 0.04793129 and for the LSTM it was 0.051983263.
As we expected, we did see the TLSTM outperform the LSTM, and if we look at the graphical results, a trend that we typically saw was the TLSTM's lag in prediction was slightly less than the LSTM's. By lag we mean that often times it seemed like the LSTM was simply trailing behind what the actual prices were, but this was not as much the case with the TLSTMs.

5.2. Analysis and Discussion

An interesting trend that we can see when analyzing the above graphs, is that although movements in closing price were often less extreme than movements in volatility, across all time intervals of 7, 30, and 90 days, our model was able to fit trends in volatility better than closing price itself.
A reason for this may have been that although stock prices are not autocorrelated, studies have shown that volatility is time varying and mean reverting (Hsieh, 1995). Intuitively then, if volatility tends to move around some n-day mean, it may be that predicting how a function moves around some value is easier to learn than some trend that is as chaotic as the closing price of a stock. With this knowledge then, it could be possible to better predict movements in price by using a model which can accurately predict volatility.
To provide an example of this, say that on day **t**, you currently own 100 shares of AAPL stock and that the actual closing price is $100, the 30 day moving average has been computed to be $150, the 30 day volatility is $50, and volatility is predicted to go down to $10. Well, assuming that our model **has** in fact learned the trends in volatility well (using the fact that mean reverting trends may be easier to predict), and it predicts volatility tomorrow to go down to $5, then it would be a fair assumption to say that tomorrow, the value of my AAPL stock is probabilistically likely to be closer to my 30 day moving average (because volatility would only go down if true closing got closer to the mean). So if we look back then, if price does mean revert towards $150 from $100, then this model would tell you to hold AAPL until tomorrow before selling.
Of course studying whether or not volatility mean reverts for *all* asset types is something that would have to be further explored, but based on previous literature and the theorems the propose and have proven, predicting future volatility could prove to be an interesting topic of study.

6. Conclusion

In our research, we surveyed a variety of pricing models and machine learning methods to see if financial modeling and forecasting were problems where ML could outperform simple financial indicators. The conclusion we reached was yes. In fact, modeling nonlinear dynamics in any system seemed to be an area where no method or approach has really gained much traction.
By primarily looking at Rose Yu's TT-RNN framework for forecasting a variety of metrics from close price and moving averages, to volatility and the market meanness index, we saw that her proposed methods do model nonlinear dynamics better than other known methods. But moving forward, more work will have to be done on how to better predict these trends.
One potential project direction that this could take, could be to incorporate the Variational Autoencoder (VAR) somehow so that the model could learn the latent characteristics within the financial data itself. Another direction this could take would be to test it on a variety of asset types. Being that we were able to only get data on Exchange Traded Funds (ETFs), it would be interesting to see how the model performs against different asset classes like Options and other Derivatives, simply because the nature of what they are and how they are traded is drastically different. Specifically, Options price prediction would be interesting because it is a known fact that the main factor influencing options pricing is market volatility. And having a model that accurately predicts volatility, *trading* in an environment where volatility is the primary driver of price movement, it would be interesting to see if prediction accuracy increases.
Additionally, building a system off of a volatility prediction model could be interesting in capitalizing off arbitrage opportunities. Being that ETFs typically have similarly priced international equivalents called ADRs, sometimes when the price of an ADR trails the price of an ETF, a tactic used by many Quantitative Funds is that they will buy the lower one (in this case the ADR), pay some set conversion fee to have it converted into the equivalent ETF, and then sell that asset to make a profit off the price difference between the two. So seeing if our model could be generalized to the case of predicting the price movement of these equivalent assets would be quite interesting as well. Because by predicting over the covariance of the two, there could be some interesting relationships.

Acknowledgements

A huge thank you to Professor Huang and all the TA's for CMSC498V for teaching such an amazing class. Although a lot of time I was lost and hopelessly confused, I appreciate your patience and dedication to helping us learn. Thank you so much!!!

References

Chen, M. Minimalrnn: Toward more interpretable and trainable recurrent neural networks. *CoRR*, abs/1704.04110, 2017. URL http://arxiv.org/abs/1704.04110.

Flunkert, V., Salinas, D., and Gasthaus, J. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1711.06788v2, 2018. URL https://arxiv.org/abs/1711.06788v2.

Hsieh, D. A. Nonlinear dynamics in financial markets: Evidence and implications. *Financial Analysts Journal*, 51(4):55–62, 1995. doi: 10.2469/faj.v51.n4.1921.

Lotter, J. C. (ed.). *The Market Meanness Index*. CreateSpace Independent Publishing Platform, 2015.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Yu, R., Zheng, S., Anandkumar, A., and Yue, A. Long-term forecasting using tensor-train rnns. *CoRR*, abs/1711.00073, 2017. URL http://arxiv.org/abs/1711.00073.