High Frequency Trading with Machine Learning Techniques

Yiqiu Luo, Zerui Ji

Abstract

The prediction of price trends to assist financial trading in the stock market is a challenging task due to the inherent complexity and dynamics in price movement. In this paper, we explore the potential of machine algorithm to help traders to make logical decisions on high-frequency trading. As opposed to lower frequency trading like daily trading or monthly trading, High-frequency trading (HFT) is a type of financial trading that takes place on the order of minutes. High-frequency decisions have to be made based on the past history, given in the form of execution prices and volumes. Our research uses Support Vector Regression(SVR) in contrast to the LSTM networks to predict the price movement of a short-terms. The experiment shows that LSTM yields better precision in all dimensions than SVR. It indicates that LSTM recurrent architecture can very effective in conducting trend prediction of stock price.

Introduction

The High-Frequency financial time series is a valuable source of information for stock market trading. Patterns hidden in the data may be used to predict the price fluctuations. The accumulated trading data provide a great stage for machine learning and another forecasting algorithm In this context Ballings, et al. (2015) [1] and Gerlein et al. (2016)[2] have obtained good results in evaluating the predictive capabilities of machine learning classifiers such as SVM, k-Nearest Neighbours (KNN), neural networks, and decision trees. The availability of high-frequency data increased the opportunities for new discoveries in the scope of machine learning, as traditional long-term stock data are subject to unexpected events and hard to apply to reality. The purpose of this research is to apply machine learning techniques for predicting high-frequency financial time series. Particularly speaking, we seek to investigate the rate of change of stock fluctuation on the short-term basis data. Due to the minimizing effect of short-term data under the expected long-term events, predictions using short-term data can lead to more independent market decisions. In this research, we use Support Vector Regression and LSTM to conduct modeling and evaluate their results with respect to prediction quality. Our research seeks to achieve the best parameter combinations for a particular regressor that yields the most optimal rate of change over a time window to help traders to make the most logical decision on trading executions.

Dataset

Our high-frequency datasets come from Lobster.com, which includes milliseconds to nanoseconds level data from four major technical companies: Amazon, Microsoft, Apple, and Intel. The datasets consist of 6 different columns: Time, Type, OrderID, Size, Price, and Direction. The time field of the raw data comes from seconds after midnight with decimal precision. The type field consists of multiple subtypes that describe the characteristics of each translation in the data. And the subtype that we are looking for is subtype 4: Execution of a visible limit order and subtypes 5: Execution of a visible limit order. The reason behind our selection is that the rest of the subtypes are related to cancellation, submission of orders, and trading halts, which are not strongly related to our focus on the rate of change of the price. We then choose to combine the size and direction. In the raw data field, '-1' means the sell limit order from the stock pool, where the +1 order means the buy action. Size is just the volume of stock being traded. Thus, through combine size and direction, we created a new column field, indicating the transacting volume. For example, if the original two fields are -1 and 300, then the new volume field will be -300. And lastly, we have the price field, which is the price of the transaction that happened at the giving time.

Solution

SVR

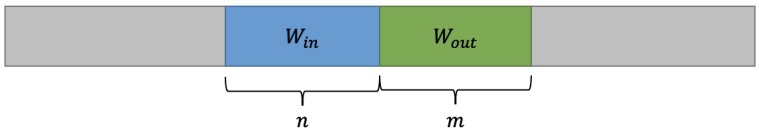

Figure 1: Data Processing

$$W_{in}(i) = [S_{i-n+1}, S_{i-n+2}, ..., S_i] \tag{1}$$

$$W_{out}(i) = [S_{i+1}, S_{i+2}, ..., S_{i+m}] \tag{2}$$

$$\overline{W}_{in}(i) = \frac{1}{n} \sum_{q=1}^{n} S_q(i) \tag{3}$$

$$\overline{W}_{out}(i) = \frac{1}{m}\sum_{q=n}^{n+m} S_q(i) \tag{4}$$

$$X_i = \frac{S_i - \overline{W}_{in}(i)}{\overline{W}_{in}(i)} \times 100\% \tag{5}$$

$$y_i = \frac{\overline{W}_{out}(i) - \overline{W}_{in}(i)}{\overline{W}_{in}(i)} \times 100\% \tag{6}$$

The essential aspect of the SVR model is that if we fix a particular moment in our price data, then our models seek to use the rate of change of the current price to the average of previous n prices to predict the average rate of change of the latter m prices. The best possible SVR model will be the combination of two sets of parameters. The first set is the hyperparameters of the model itself, and the second set is the size of windows to calculate the average rate of change. Thus, to prepare the modeling of SVR, there is some transformation that has to be done and mathematical notations to be formally stated. Denote S_i as raw stock price, W_{in} as the average of previous n stock price data from the index i, shown in the equation(3). Then the new rate of change X_i can be calculated through equation(5). And Y_i, the ground truth rate of change to be calculated through equation(6)

In order to get the most optimal model, it is necessary to perform hyperparameter tuning on different choices of kernels based on fixed window size. Arbitrarily, we choose to use $(n, m = 60)$. Through comparing the MAE(mean absolute loss) and MSE(mean square loss), the final model that we choose is the linear SVR, having the smallest loss in both criteria: 0.0172 and 0.00059. It is interesting to observe that this actually resembles the linear characteristics of time series stock data. Penalty Parameter C and tube parameter Epsilon are also altered to achieve the max performance. After comparing the results generated by different combinations of C and Epsilon, our conclusion reaches that the penalty parameter C does not significantly affect the performance quality. However, it reduces the time for the model to complete. The tube parameter ϵ improves the regressor's performance by a small order but increases the time spent significantly.

After the model has been tuned to the most optimal value based on fixed window size, we then moved on to find the best possible window size for the optimal model. The results of the different m and n can be found in the section SVR vs LSTM model results analysis.

LSTM

High-frequency stock trading data is well known for its nonlinearity and low signal-to-noise ratio. These properties make high-frequency stock trading data perfect in deep learning studies. In recent years, there are many successful attempts in applying LSTM in the stock market. For example, in 2020, Jiayu

Qiu and his team [3] showed that LSTM with denoised input stock trading data could quite accurately predict stock prices. To apply LSTM to our high-frequency data, we applied the methodologies below.

The nature of LSTM modeling requires the previous n input to be remembered and studies to predict the next output. And if we keep following the data transformation in the SVR model design, preprocessing all the actual prices to the rate of change before pipelining the input into the model, this can lead to a serious weights redundancy problem.

Referring to how we process the rate of change before. Each X_i is calculated through S_i/W_i (avg sum of all previous n indices), meaning that X_i is directly related to S_{i-n} to S_{i-1} and X_{i+1} is related to S_{i-n+1} to S_i. It is true that this is easier to compute since all the input is the rate of change. However, we find that this would result in a highly correlated chain of input, as LSTM will take all these correlated inputs into memory. Resulting in the occurrence of each price index being counted differently. For example, imaging a window of $[-n, n]$, and a middle point of S_i. To calculated X_i to X_n, the S_{i-n} will be counted one time, the S_{i-n+1} will be calculated two times, so on and so forth, resulting in the S_{i-1} being counted n times. Thus, if we continue to use the previous data processing to transform the S_i into X_i and then input the n of the rate of change. Not only is each input cell not independent, but the prices index at the middle will also occupy a much bigger weight to the model.

Thus, we decided to use prices directly and put them into the LSTM model. Notice that the defaults of the LSTM model use *tanh* as the activation function. We believe that this is a precise depiction of real stock trading. A negative stock price should have an impact on the prediction as well. This naturally inclines us to use *tanh*, which has an output between -1 and 1. Since the scale of some of the features is not proportional to others, to achieve a more steady result, we applied batch normalization to rescale the data. And after the predicted price is calculated, it is put into the rate of change.

Accuracy

Comparing to image recognition or speech processing, it is often hard to define what is correctly predicted in the financial world. Rather, accuracy is often a relative term. Prediction of the exact numeric figure is certainly a top-notch thing to have. Yet, from a relative point of view, +0.45 and +0.458 might just have a similar realistic meaning to any trader. Thus, we define the following rules for accuracy. First, the direction matters, if our model is able to predict the correct trend for meaning, the same + and - sign, then we will classify this as a correct prediction. However, the combination of +0.5 and +0.1 between test and prediction is obviously different between +0.5 and +0.4, as the former combination should bear more weight. Finally, we have this weighted equation for accuracy in the same direction. $ACC_{direction} = 1 \times (1 - abs(Y_{predict} - Y_{test})$. Secondly, in the real-world scenario, we should also take into account results that are in the opposite direction yet having very small numeric differences.

Thus, if $(Y_{predict} - Y_{test})$ have a numeric difference of less than 0.05. This would also be classified as correct.

Results and Analysis

SVR

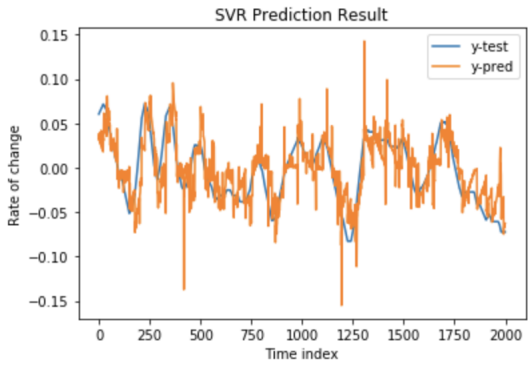

Figure 2: SVR Result

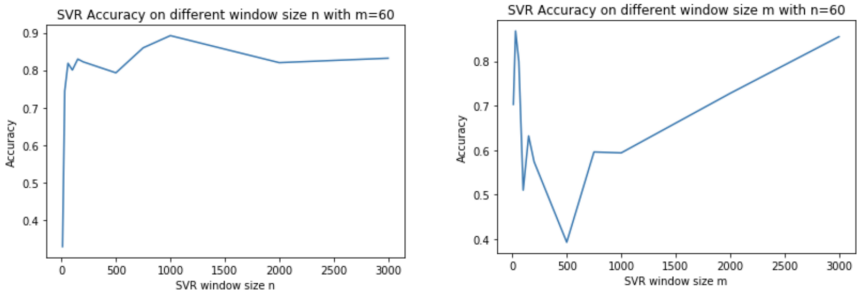

Figure 3: Tracing

Figure 3 are traces of the rate of change under the condition of our optimal tuned SVR model and window parameters n and m. The model under Figure 2 gives a relatively good estimate of the ground truth rate of change with 0.83 accuracy rate and the trend is correctly captured. With this model as our baseline, we start to experiment with the combinations of n and m onto our model. Before getting the details of Figure 3, there are a few acknowledgments that should be made. We do not have enough computational resources available to run every single n and m combination. Thus we decide to root the variation

of parameters n and m from our optimal SVR model of $(60, 60)$ and inspect their influence over it. The first graph is a demonstration of the influence of different n on the model by fixing $m = 60$. We see that the maximal accuracy reaches at $n = 1000$, meaning to include previous 1000 prices for calculating the average rate of change. Since the datasets are milliseconds to nanoseconds level, 1000 prices are roughly about 30s in the real life. The accuracy here is about 0.91. The second graph fixes on n and varies m window size(number of transaction prices after fixed price point. Notice that the accuracy drops dramatically until $m = 500$ and then has a straightforward increase afterward. This is a direct reflection of the redundancy problem previously discussed. We see both a drop of accuracy sample in the SVR cases by including more and more data sets. Moreover, the drop on m is much more severe, which is caused by the insignificant data input of n. With only 60 data inputs, it is hard to give an accurate rate of change over a range of 500 results. Furthermore, though the accuracy rate increases, as m further grows, it is due to the macro influence of stock trend, instead of being more accurate on the micro level.

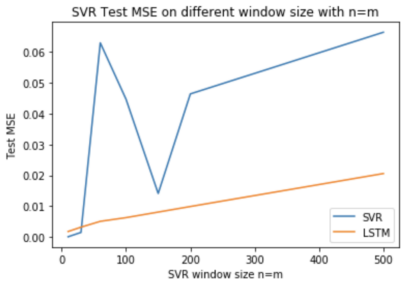

Figure 4: SVR, n and m

LSTM

Comparing to SVR, LSTM has a much more consistent picture of accuracy. In the first graph, we see that by having more input data, the accuracy can achieve an incredibly high rate. However, there is not a huge increase in accuracy that is brought by more inputs.

In the Figure 5, we also see an increase in accuracy by including more data(vary n) when calculating the rate of change. We still suffer the curse of macro-influence of the entire stock's trend, as the graph of test MSE shown above for both SVR and LSTM. The model suffers more and more MSE loss when we include more stock transactions into the model prediction. This can result in highly volatile predicted results when this model is applied to other datasets, which would not be ideal. Thus, in the LSTM case, the best range of n and m lies between $(100, 150)$ transactions. In real-world time units, this is

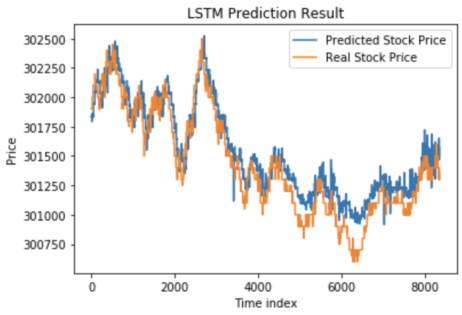

Figure 5: LSTM Result

between $(3, 4)$ seconds. The accuracy rate is about 0.9899. The results of our LSTM optimal model tracing are shown in Figure 5.

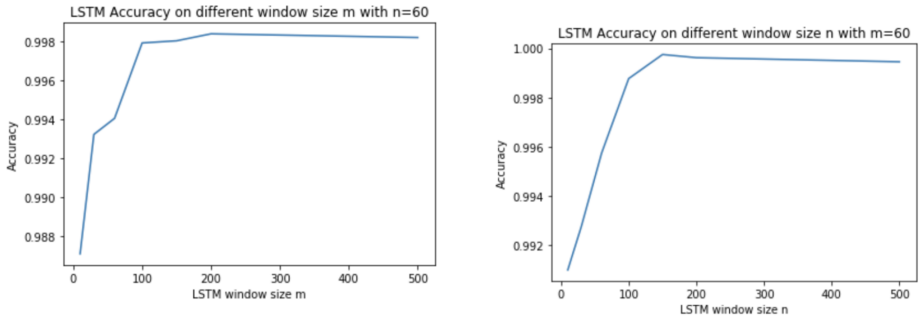

Figure 6: LSTM Tracing

Conclusion

In this paper, we focus on using Support vector regression and LSTM to predict the rate of change in High-Frequency Trading scenarios. The comparison of two different models proves to us that LSTM is more effective in making logical decisions. Moreover, through experimenting on different window variables, we are able to find the most optimal range for the combinations of n and m under a reasonable amount of MSE error. We have also acknowledged the shortcoming of our model as it is still under the influence of macro-level stock trending. The influence is more evident as we incorporate more and more input data in our sliding window.

Future work

Firstly, we can consider using a limit order book of multiple levels instead. For our study, we focused on level one which is consists of the price and volume that is actually being executed. Models with multiple layers can have additional information and features to predict the rate of change.

Secondly, We also want our data to more comprehensive. Because these four companies have the same trend since all of them are technology companies. They appeared to have the same consistent macro-level downward trend. This makes the model easier to approximate with a higher accuracy rate, especially including more data into our n,m parameters. Therefore, more execution data on multiple days, multiple trends, multiple industries are needed.

Last of all With more time and computational resources, more comprehensive combinations of n and m can make the model more detailed and accurate. And actually considering building a trading bot to use our n and m parameter to automatically trade the in real-world stock market

References

[1] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating multiple classifiers for stock price direction prediction," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7046–7056, 2015.

[2] E. A. Gerlein, M. McGinnity, A. Belatreche, and S. Coleman, "Evaluating machine learning classification for financial trading: An empirical approach," *Expert Systems with Applications*, vol. 54, pp. 193–207, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416000282

[3] B. W. Jiayu Qiu and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism." *Plos One*, vol. 15(1), 2020.