CS230

Using LSTM in Stock prediction and Quantitative Trading

Zhichao Zou
Center for Professional Development
Stanford University
SUNetID: zzou
zzou@stanford.edu

Zihao Qu
Center for Professional Development
Stanford University
SUNetID: quzihao
quzihao@stanford.edu

Abstract

The direction of the financial market is always stochastic and volatile and the return of the security return is deemed to be unpredictable. Analysts now are trying to apply the modeling techniques from Natural Language Processing into the field of Finance as the similarity of having the sequential property in the data. In this research, we have constructed and applied the state-of-art deep learning sequential model, namely Long Short Term Memory Model (LSTM), Stacked-LSTM and Attention-Based LSTM, along with the traditional ARIMA model, into the prediction of stock prices on the next day. Moreover, using our prediction, we built up two trading strategies and compared with the benchmark. Our input data not only contains traditional end-day price and trading volumes, but also includes corporate accounting statistics, which are carefully selected and applied into the models. The accounting data are regarded as the news and price shocks to a corporation, hence are expected to increase the predictive power of the model. The result has shown that Attention-LSTM beats all other models in terms of prediction error and shows much higher return in our trading strategy over other models. Furthermore, we discovered that the stacked-LSTM model does not improve the predictive power over LSTM, even though it has more complex model structure.

1 Introduction

In the field of quantitative trading, predicting the future security returns lies in the center of the industry, as the future trading strategy is always deployed and created based on our view of the financial market in the future. The trading area has two main different methods, namely fundamental analysis and quantitative trading. Fundamental method makes the trading decision based on the subjective view on the industry or company's future direction, it mainly relies on the public information such as market news, corporate statistics as well as a series of financial statement releases. On the other hand, the quantitative trading strategy uses mathematical models to make the decision hence avoids the interruption of human subjectivity and emotion. Traditionally, the methodology of quantitative strategy involves using linear regressions, ARIMA model as well as GARCH model to capture the features of time series and the stochasticity of the volatility. These methods were proved to be effective for a certain period of time in the old regimes. As the regime shift happens in the financial industry, these models became less effective. The quantitative trading industry has therefore shifted into the 'deep learning era', which has been more frequently used nowadays.In this research, we predicted stocks return using the deep learning model, more specifically LSTM and Attention-LSTM models. Conventional financial time series prediction only uses price and volume to predict the future

CS230: Deep Learning, Winter 2020,

prices. In this work, the input includes the usual price and volumes, as well as the corporate statics. Usually these statistics are released quarterly by the companies and have significant impacts on future price movements. The output of our model is the price or scaled price on the next day. Once the future price is predicted, we will build up a quantitative trading strategy based on the prediction. The return of our strategy is compared with the market.

2 Related Work

During the pre-deep learning era, Financial Time Series modelling has mainly concentrated in the field of ARIMA and any modifications on this, and the result has proved that the traditional time series model does provide decent predictive power to a limit. For example, due to the asymmetric distribution in financial time series return, Minyoung Kim has replaced the traditional Maximum Likelihood Estimation with an asymmetric loss function.[7] C.K. Lee et al. compared the forecasting performance of ARIMA and artificial neural networks on Korean stock price index. The work showed that ARIMA provided more accurate forecasts than the back-propagation neural network.[8] More recently, deep learning methods have demonstrated better performances thanks to improved computational power and the ability of learning non-linear relationships enclosed in various financial features. Sreelekshmy Selvin et al. compared three different deep learning architectures including RNN, LSTM, and CNN-sliding window models for the prediction of NSEI listed stocks.[11] They concluded that CNN architecture is capable of identifying changes in trend of stocks and outperforms other models. Yan and Ouyang combined the wavelet transform of the financial time series with the LSTM and showed that the resulting model beat the performance of traditional Support Vector Machine, and K-nearest Neighbours.[12] Thien Hai Nguyen et al. demonstrated that the integration of sediment features extracted from social media can improve the accuracy of prediction.[10] The performance of LSTM-RNN will be further boosted by feeding relevant data based on financial domain knowledge. [3, 5] Moreover, Kim Won has developed a hybrid approach to combine LSTM and GARCH models and the resulting model has much lower prediction errors. [6]

3 Dataset and Features

The data we utilized to train/develop and test our model include two aspects: 1. The daily prices and volumes for every SP 500 stock from 2004 to 2013. 2. The accounting and corporate statistics for the SP 500 stocks from 2004 to 2013. Two sets were merged by date and forward filled missing statistics between two releasing date. After large number of iterations in selecting the appropriate input, the input we use are 'adjustment close price', 'trading volume', 'Debt-to-Equity Ratio', 'Return on Equity', 'Price-to-Book' ratio, 'Profit Margin', 'Diluted Earnings Per Share' and 'Company Beta'. We applied the min-max scale to normalize the data between 0 and 1, hence prevent the magnitude of certain features overwhelms others. For each stock in SP 500 we have the daily data mentioned above from 2004 to 2013. We use the approximate ratio of 70-15-15 to split the data for each stock in training, development, and testing data. In other words, we use data from 2013 to 2011 as training data, 2012 as development data and 2013 as testing data. Our performance metrics and trading strategies are hence built on the data on 2013. A sample piece of our data is shown Figure 1.

date	adj_close	volume	DE Ratio	Return on Equity	Price/Book	Profit Margin	Diluted EPS	Beta
2013-12-17	0.950181	0.019849	0.652485	0.01414	0.046009	0.626554	1.0	0.24031
2013-12-18	0.964769	0.036554	0.652485	0.01414	0.046009	0.626554	1.0	0.24031
2013-12-19	0.966209	0.023068	0.652485	0.01414	0.046009	0.626554	1.0	0.24031
2013-12-20	0.980317	0.062588	0.652485	0.01414	0.046009	0.626554	1.0	0.24031

Figure 1: Data structure of Google stock price and corporate accounting statistics, from 2004 to 2013

4 Methods

In order to examine the impact of predictive powers in different financial Time Series, we built three deep learning models as well as one traditional time series model. They are 1) Time Series Model (ARIMA);2) RNN with LSTM Model (LSTM); 3) RNN with Stacked-LSTM (Stacked-LSTM);4) RNN with LSTM + Attention (Attention-LSTM).

Time Series Model: AutoRegressive Integrated Moving Average (ARIMA) model is a widely used statistical method for time series forecasting (equation 1). In this work, we followed the Box-Jenkins Methodology to build an ARIMA model as a baseline to compare with Deep Learning models. [4] For the ARIMA model, only "adjusted close price" was used to fit the model. We used summary statistics and functions such as moving average and autocorrelation function to identify data trends and the parameters (p, d, and q) of ARIMA model.

$$\delta Y_t(p,d,q) = \mu + \sum_{p=1}^{p}(\phi_p \times \delta Y_{t-p}) - \sum_{q=1}^{p}(\theta_q \times e_{t-q}) \quad where \quad \delta Y_t = Y_t - Y_{t-d} \tag{1}$$

RNN with Single/Stacked-LSTM: The main idea of RNN is to apply the sequential observations learned from the earlier stages to forecast future trends. Long-Short Term Memory (LSTM) model is an updated version of RNN. It can overcome the drawback of RNN in capturing long term influences.

LSTM introduces the memory cell that enables long-term dependency between time lags. The memory cells replaces the hidden layer neurons in the RNN and filters the information through the gate structure to maintain and update the state of memory cells. The gate structure includes input gate, forget gate and output gate.

The forget gate in the LSTM determines which cell state information is discarded from the model, it accepts the output from the previous time step h_{t-1} and the new input of the current time step. Its main function is to record the amount of information reserved from the previous cell state to the current cell state. It will output a value between 0 and 1 where 0 means complete reservation and 1 means complete abandonment.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

The input gate determines how much the current time network input x_t is reserved into the new cell state C_t, it avoids feeding the unimportant information into the current memory cell. It has three different components: 1) Get the state of the cell that must be updated; 2) Create a new cell state; 3) Update the cell state to the current cell state.

$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i) \quad \hat{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{3}$$

The output gate controls how much the newly created cell state will be discarded, the output information is firstly determined by a sigmoid layer, then the newly created cell state is processed by $tanh$, together with the sigmoid output to determine the final output.

$$O_t = \sigma(W_\sigma \cdot [h_{t-1}, x_t] + b_o) \quad h_t = O_t * tanh(C_t) \tag{4}$$

Due to the higher stochasticity of financial time series, we will build up two models in LSTM and compare their performances: one single Layer LSTM memory model, and one Stacked-LSTM model. We expected the Stacked-LSTM model can capture more stochasticity within the stock market due to its more complex structure. However, our experiments showed the opposite results, and we will discuss below.

Attentions LSTM: Machine learning algorithms are inspired by biological phenomena and human perception. For instance, we do not treat all information with equal importance, instead human perception focuses on the important parts first for the newly received information. This phenomenon is analogous to the financial market as well, as the prices of securities assign different levels of importance into the market information, and it prompts us to use the Attention Mechanism to add this feature into our RNN LSTM. In our model, we apply the soft attention, where we updated the input of the model by assigning weights to input information based on the learning results and obtaining results in a more logical order. Mathematically, it is formulated as:

$$e_t = tanh(W_a [x_1, x_2, ..., x_T] + b) \quad \alpha_t = \frac{exp(e_t)}{\sum_{k=1}^{T} exp(e_k)} \tag{5}$$

Where w_a is the weight matrix which is a trainable parameter, it indicates the amount of information that should be emphasized. The resulting weighting is α_t, which will be used to weight the input. The original input will be replaced by the newly weighted input and is used for updating the attention. This new input can pay more attention to the specific input feature sequence, extracting the key feature effectively and ignoring the redundant features using the attention weights. This is the process of converting the original LSTM model into an attention based model. We use the framework setup by Qianqian for the Attention-LSTM and updated it to fit for financial models.[2, 1] Theoretically we expect to see better model performance for Attention-LSTM than LSTM.

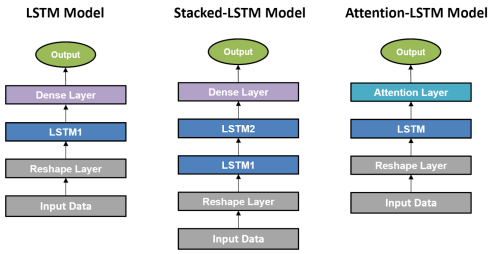

Figure 2: The architecture of three deep learning models

5 Experiments/Results/Discussion

In this project, we will be using data from the past to predict the return on the next trading day. To assess the model, our primary model performance metric is Mean Squared Error (MSE). The MSE will be performed on the test data, which is totally unseen from the training and development stages. It is calculated between our predicted price and the true price.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{6}$$

The secondary model performance metric is related with our trading strategy. We build up two intraday trading strategies based on our prediction of performance in 2013:

- Long-Only Strategy: For the prediction for each stock on the next day, if the prediction is positive, we buy the stock at the open price and sell the stock at close price in the same day. If the prediction is negative, no action is taken.
- Long-Short Strategy: For the prediction for each stock on the next day, if the prediction is positive, we buy the stock at the open price and sell the stock at close price in the same day. If the prediction is negative, we short-sell the stock at the open price and close out the short-sell at the close price.

Notice that our trading strategy is an intraday trading strategy, meaning that we do not hold positions overnight, so we only take actions on the same day.

Our initial sample contains the entire universe of 500 stocks in SP. However, after some experiments, we decided to not use all 500 stocks and switched our gear to the top 10 stocks in SP 500 based on market capitalization. As greater market capitalization usually implied more stable financial status, more predictable growth and more complete public data from the companies.

For the LSTM models, the $tanh$ activation function was used for the LSTM layers and the $sigmoid$ activation function was used for the output layers, these activation functions have been proved to be showing more superior results than other activation functions. Moreover, the dropout probability of 20% is also applied for each hidden layer as the usual regularization method to reduce the overfitting issue. Finally, the adam optimization is used for learning the parameters, and the mean squared error is utilized as the loss function.

To increase our training speed and increase the chance of convergence, we use mini-batch in our training process. We have attempted a large number of mini-batch sizes and eventually decided to use the size of 30, as greater size does not guarantee convergence and sometimes trapped in the saddle point, and smaller sizes makes the training process quite slow. The lookback day is the numbers of days we use from the past to predict the future, in this research we have experimented a large number of look back days and eventually decided to use look back days of 20 days, meaning that we will be using the data from past 20 days to predict the stock prices in the next day.

ARIMA model: The parameters (p, d, q) of ARIMA model were determined to be (1, 0, 1) based on the MSE on the training set. This actually makes the model an ARMA model, where no differencing was performed to change the stationarity. The MSE on the test set of Google's data was 0.0546. However, the model gave a linear prediction from test data, even the forecasted trend is opposite to the real price trend. The results suggested ARIMA model did not perform well in predicting non-linearity and long-term prediction. Hence, this model was not included in the evaluation of annual return.

Deep Learning Models: Each model was trained by 100 epoches for each stock, and the MSE for the testing sample (which is the 2013 data) is performed on each of 10 stocks. Comparison of the results for each model is shown below in Figure 3. Clearly the Attention-LSTM performs better than both the Stacked-LSTM and LSTM models as expected. As for long-term time series prediction, the Attention-LSTM brings advantages of selecting the important and relevant information hence enhancing the predictive accuracy. For certain stocks where it experienced large fluctuations in the year such as Google and IBM, the attention model beats the other two by quite a big margin, further emphasizing the significance of relying on the context to predict the stock.

Typically, the deeper neural network is able to explain more complicated problem than single-larger neural network. However, another interesting finding we discovered is the fact that the stacked-LSTM does not significantly outperform the LSTM in the context of stock price prediction. Instead, the performance LSTM even beat the stacked-LSTM in certain instances. It has proved that the more complex representative does not necessarily improve the predictive power. It is possibly due to two reasons: 1. The more complicated neural network representation causes overfitting issue, the greater number of parameters in stacked-LSTM memory model does not generalize well in the unseen data. 2. The stacked-LSTM is more suitable in predicting classification problems rather than continuous time series like stock prices

Due to the suboptimal performance in the stacked-LSTM model, the trading strategy is only built on the LSTM and Attention-LSTM models. The return of our trading strategy in Figure 4 is the sum of the individual returns from each of the 10 stocks above. The benchmark return is used for comparing the performance of the strategies and is set as the 2013 annual return of SP 500. The trading strategy based on our return prediction beat the beachmark by quite a big margin, reflecting the accuracy of the prediction from LSTM and Attention-LSTM models. Moreover, the annual return from the Attention-LSTM model beat the LSTM model, further confirming the better predictive power from Attention-LSTM model. As the attention model is able to analyze the context of the past and select the most relevant information for the prediction, it is therefore better in deciding the long and short signal.

Mean Squared Error ($*10^{-3}$)			
Stock Ticker	**LSTM**	**Stacked-LSTM**	**Attention-LSTM**
XOM	25	25	11
GE	18	2	3
WMT	128	163	76
MSFT	40	27	16
IBM	28	7	6
AAPL	54	61	44
GOOG	35	37	26
GS	12	16	18
PFE	32	64	49
JNJ	57	123	41

Figure 3: MSE summary for three deep learning models

Trading Strategy Annual Return			
Annual Return	**LSTM**	**Attention-LSTM**	**Benchmark**
Long-Only Strategy	173%	266%	30%
Long-Short Strategy	99%	263%	30%

Figure 4: The return of trading strategy based on LSTM, Attention-LSTM and Benchmark

6 Conclusion/Future Work

This paper establishes a forecasting framework to predict the prices of stocks. We leveraged the combinations of price, volumes and corporate statistics as input data. We proposed, developed, trained and tested four models: ARIMA, LSTM, Stacked-LSTM and Attention-LSTM models, and built up Long-Only and Long-Short trading strategies according to our model predictions.The attention-LSTM shows more superior results over other models due its its ability to assign different weights to the input features hence automatically choose the most relevant features. Hence the Attention-LSTM is more able to capture the long-term dependence in the time series and more suitable in predicting financial time series. Our superior trading return from the Attention-LSTM further validates our experimental result. Moreover, we have shown that despite the more complicated model structure of stacked-LSTM over single LSTM model, the stacked-LSTM does not have better model performance over the single LSTM model due to the potential of overfitting. One direction of future work will be dealing with the volatility of stock time series. One difficulty of predicting stock market arises from its non-stationary behaviour.[9] It would be interesting to see how Attention-LSTM performs on denoised data.

7 Contributions

Zhichao Zou and Zihao Qu all collectively and equally contributed to the research, development of our four models, reporting writing, poster creation, and Zihao Qu is responsible for the video presentation.

References

[1] Framework used: Keras, tensor flow.

[2] Toxic comment classification challenge, https://www.kaggle.com/qqgeogor/keras-lstm-attention-glove840b-lb-0-043/log.

[3] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[4] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[5] JB Heaton, NG Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

[6] Ha Young Kim and Chang Hyun Won. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103:25–37, 2018.

[7] Minyoung Kim. Cost-sensitive estimation of arma models for financial asset return data. *Mathematical Problems in Engineering*, 2015, 2015.

[8] CK Lee, Y Sehwan, and J Jongdae. Neural network model versus sarima model in forecasting korean stock price index (kospi). *Issues in Information System*, 8(2):372–378, 2007.

[9] Hieu Quang Nguyen, Abdul Hasib Rahimyar, and Xiaodi Wang. Stock forecasting using m-band wavelet-based svr and rnn-lstms models. *arXiv preprint arXiv:1904.08459*, 2019.

[10] Thien Hai Nguyen, Kiyoaki Shirai, and Julien Velcin. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603–9611, 2015.

[11] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

[12] Hongju Yan and Hongbing Ouyang. Financial time series prediction based on deep learning. *Wireless Personal Communications*, 102(2):683–700, 2018.