Online Long Short-Term Memory Network for Stock Trading

Dušan Fister
Univerza v Mariboru, Ekonomsko-poslovna
fakulteta,
Razlagova 14,
SI-2000 Maribor, Slovenia
dusan.fister1@um.si

Timotej Jagrič
Univerza v Mariboru, Ekonomsko-poslovna
fakulteta,
Razlagova 14,
SI-2000 Maribor, Slovenia
timotej.jagric@um.si

ABSTRACT

Economic theory teaches that it is impossible to earn higher-than-normal returns on trading stocks, and that the only chance to earn higher profits is to take higher risk. Often, the practice reveals that higher-than-normal stock returns may indeed be earned, which is a hypothesis we would like to empirically test. In this way, we design so-called mechanical trading system, which focuses on the technical analysis of past stock data, and perform daily trading for the German stock from period 2010-2019. Long short-term memory network is taken as a basis of the mechanical trading system. Obtained results show that higher-than-normal returns are easily obtainable, which polemizes the efficiency of the observed stock.

Keywords

LSTM networks, mechanical trading system, stock trading

1. INTRODUCTION

Efficient market hypothesis (EMH) states that asset prices fully reflect all information available [3] and that asset prices quickly incorporate any new information without delay. Consequently, higher-than-normal returns cannot be achieved and predictions of future prices cannot be valuable [10]. Stocks strictly follow a random walk (are unpredictable) and it is impossible to beat the market. Malkiel [10] reports opposite opinion, where he states that market inefficiencies and arbitrage opportunities to earn higher-than-normal returns indeed exist. A sample of these includes market bubbles, market crashes and other irrational pricing. Pedersen [12] even outlines common investment strategies to detect arbitrage opportunities. To investigate any market inefficiencies for a case study of Germany, we employ a single-stock automated mechanical trading system (MTS). As a benchmark, we choose the stock Daimler AG. We implement the MTS trading strategy using the long short-term memory network (LSTM) and compare its performance to the pas-

sive trading strategy. LSTMs are found usable in broad areas, such as sentence classification [2], trajectory prediction of autonomous vehicles [9], flood forecasting [8] and malware detection [15]. Furthermore, LSTM are used in engineering for estimating remaining useful life of systems [17] and in medicine for automated diagnosis of arrhythmia [11]. The structure of the paper is as follows: chapter two outlines the fundamentals of LSTM networks. Chapter three lists the information about the dataset and explains methodology. Chapter four shows the experiments and results, while chapter five concludes the paper.

2. LONG SHORT-TERM MEMORY

Long short-term memory networks (LSTMs) are a kind of artificial neural networks, specifically designed to deal with sequential and dynamic data [6]. LSTMs are recurrent neural networks (RNN) with an agile structure that attempt to remember long-term dependencies and prevent both the usual RNN problems: exploding and vanishing gradients. The benefit of the LSTM lies in a memory cell, i.e. central element of the network, and the three kinds of gates which control the flow of information. Memory cell accumulates the internal state by supplying the sequences of data. Each gates constitutes of weights, which are adapted during the learning process [5]. Figure 1 shows the structure of the usual LSTM. The common LSTM structure consists of the

Figure 1: LSTM structure.

forget, input and output gate. Forget gate, which is a simple *sigmoid* function, is used to filtrate past information to be brought into following time step. Input gate, which is a combination of *sigmoid* and *tanh* functions, is used to enter new information into the memory cell. Output gate,

which is also a combination of *sigmoid* and *tanh* function, is used to output relevant information from the memory cell and transfer the information into following time step. LSTM layer can be single, or stacked into multi-layered networks. Figure 1 shows the individual LSTM layer, which is sufficient for many cases. For more comprehensive (e.g. non-stationary) data, [1] suggests the use of stacked LSTM network, e.g. two individual LSTM layers composed into a single network.

Table 1: Dataset variables, used during trading.

Explanatory variables	Vars.
1. Stock and market data: open, close, high, low, adj. close, volume	12
2. Date data: month, day, day of week, days to next trading day, days from previous trading day	5
3. Technical indicators: RET: $n = \{1, 2, 3, \dots, 10\}$ -day period DIFF: $n = \{1, 2\}$ -day period DIFF.RET: $n = \{1, 2\}$ -day period RSI: 14-day period RSI MACD: 12-day short, 26-day long and 9-day signal period INCL: $n = \{5, 10, 15, 20\}$ -day period INCL.CHG: REL.DIFF:	10 2 2 1 1 4 4 2
Sum	43

3. MECHANICAL TRADING SYSTEM

LSTMs can be applied to diverse problems and domains of sequential data. In this paper, we apply the LSTM for classification problem, which drives the MTS. MTS follows the underlying daily stock prices and outputs the three common trading decisions respectively: buy, hold and sell. According to the trading decisions, stocks are bought, sold or held in the portfolio. Since the movement of stock prices affects the portfolio value, it is desired that the current and final value of portfolio (and thus profit), is maximized. For implementation of MTS, we assume (1) perfect liquidity of stocks and (2) close price trading. Perfect liquidity means that stocks can be sold or bought instantly. Close price trading means that stocks are traded at the close price, just after the market close. The MTS trades with stocks daily. Here, only the relevant historic daily trading data is supplied. No additional information, such as company's financial reports and other statements, are taken into account, since these are available for longer time periods only, e.g. a quarter of a year. The LSTM is thus able to extract daily trading patterns and identify trading opportunities that arise by behavioural acting of ordinary investors. These are often psychologically-affected and subjected to phenomena, such as panicking and following the herd instinct. Those two largely drive the price of a stock and forecasting them may be beneficial to earn higher-than-normal returns.

3.1 Dataset

Daimler AG is a German automotive company, with approximately 50 B Eur of market capitalization, 1.07 B of shares outstanding and almost 4.6 M of trades daily, at the time of

writing. It is one of the blue-chip stocks, listed in the index of 30 most prominent German companies DAX30. The stock data is obtained from Yahoo Finance website¹, during the period from 4 Jan 2010 - 7 Jun 2019 on a daily basis. Following data is downloaded: the highest daily price (*High*), the lowest daily price (*Low*), open price (*Open*), close price (*Close*), daily volume (*Volume*) and adjusted close price (*AdjClose*). The data is downloaded for the Daimler AG stock and DAX30 industrial index. Composing those two into a dataset, several other variables are derived from. In this way, a more expanded, detailed and concrete dataset is built. Table 1 lists the dataset variables. There are 12 stock and market data variables, 5 variables which symbolize the date and 8 technical indicators. 10 stock returns (RET) are extracted, ranging from 1-day returns to 10-day returns. 2 stock differences (DIFF) are obtained as differences between today and yesterday's price and today's price and price 2 days ago. Difference of returns (DIFF.RET) is calculated by subtracting the stock returns. Relative strength indicator (RSI) and moving average convergence divergence (MACD) are common trading indicators. Inclination (INCL) is obtained by calculating the regression coefficient over n periods of stock prices. Inclination change (INCL.CHG) is calculated from the 5-day inclination and the 5-day inclination five days ago and current 5-day inclination to current 10-day, 15-day and 20-day inclinations. Relative differences (REL.DIFF) are obtained as ratio between close and open prices, and high and low prices. Response variable ($\mathbf{y}$) is generated by deriving the 1-day stock returns. If the underlying stock return overcomes the fixed threshold, set at 0.01 (1%) in our case, the response variable is marked as *Buy*. Furthermore, if the stock return exceeds the negative value of threshold, it is marked as *Sell*. If it does not exceed any of the two, it is marked as *Hold*. These signals are encoded in binary form to fulfill the requirements of the dense layer.

3.2 Methodology

The work with LSTM is divided into two samples, i.e. training and prediction, since LSTM is a modeling tool and thus needs some data to be trained from. Initially, the training sample of the LSTM is used for offline training and is intended to capture general long-term stock dependencies. It is performed only once. Additionally, an online learning and regular re-train of the LSTM is applicable to update the pre-built LSTM model with the latest information, i.e. latest stock price movements. The procedure of training and prediction samples is explained in Algorithm 1. First the stock data (dataset) is normalized in range(0,1) and batch size b is defined. Dataset is split into two samples. In-sample is used for the offline training and after, the memory cell is reset. The LSTM is used to predict the in-sample and establish a stable memory cell. Predictions on the in-sample are biased and are not used for real trading. After the internal state is established, the out-of-sample is used for prediction. Since stacked LSTMs require the batch size to be divisible from the number of samples, the first b -samples from out-of-sample are only used for prediction each time. Its solutions are appended into a vector, to be evaluated later. The iterative online training comes next, where each time, the b -samples are transferred from out-of-sample to in-sample. By expanding the in-sample for b -samples, out-

¹<https://finance.yahoo.com>

Algorithm 1 Algorithm of online LSTM training.

```

1: procedure ONLINE LSTM TRAINING
2:   NORMALIZE(dataset);
3:   DEFINE  $b$  ▷ define batch size
4:   in-sample, out-of-sample = SPLIT(dataset);
5:   for no. of epochs do ▷ number of epochs
6:      $model \leftarrow$  TRAIN(in-sample); ▷ offline training
7:   end for
8:   RESET_STATE( $model$ ) ▷ reset memory cell state
9:   PREDICT(in-sample) ▷ establish memory cell state
10:   $decision \leftarrow$  PREDICT(out-of-sample[0:b])
11:  while out-of-sample do
12:    in-sample = APPEND(out-of-sample[0:b])
13:    out-of-sample = REMOVE(out-of-sample[0:b])
14:    RESET_STATE( $model$ )
15:    for no. of epochs / 100 do
16:       $model \leftarrow$  RETRAIN(in-sample); ▷ online
17:    end for
18:    RESET_STATE( $model$ )
19:    PREDICT(in-sample)
20:     $decision \leftarrow$  PREDICT(out-of-sample[0:b]) ▷
21:  end while
22:   $results \leftarrow$  BACKTEST( $decision$ )
23:  INTERPRET_RESULTS
24: end procedure

```

of-sample is simultaneously reduced for the same amount. The LSTM pre-built model is then retrained using the expanded in-sample, which practically means that b trading days need to (physically) pass prior any LSTM online retrain occurs. For retraining, a lower number of the number of epochs is taken. The memory cell is next reset and the expanded in-sample predicted to establish internal state of the memory cell. Next, the first b -samples of the remaining out-of-sample are predicted and its solutions appended in the **decision** vector.

The quality of the MTS trading strategy, i.e. **decision** vector, is evaluated using the back-testing approach. By back-testing, the MTS is put into history and is given an initial amount of cash, which is used to buy stocks during the trading. MTS follows underlying movement of stock prices day-by-day and trades with stocks in a continual trading process. If the LSTM supposes that tomorrow's close price of a stock will increase over the preset threshold, it gives the signal to buy. Alternatively, if the price is expected to drop below the threshold, the MTS gives the signal to sell. Once the signal to buy (*buy*) is classified, maximum number of stocks, reduced for transaction costs, are bought for the current amount of money. Similarly, when the signal to sell (*sell*) is classified, all the stocks are sold and the amount is lowered for transaction costs. No information about any forthcoming stock prices is specified at any time.

4. EXPERIMENTS AND RESULTS

The goal of the experimental work is to show that automated MTS using the LSTM trading strategy can be used to catch excess profits (higher-than-normal returns) on trading stocks. Daimler AG stock is taken as a benchmark. Its dataset is split 70%-30%, where 70% goes to in-sample and

the rest 30% to out-of-sample. Framework is implemented in the Python programming language, using the Keras Deep Learning Library². A two layered stacked LSTM and Adam optimizer are used [7]. Figure 2 outlines the structure of the used network, while the table 2 the algorithm setup. Trading costs of 1% are adopted, according to literature [13, 14, 16], which come into play when buying or selling stocks.

Figure 2: Structure of the network.

Experiments are conducted using the out-of-sample back-testing. The higher the final portfolio value, the better the trading strategy. Results are reported graphically. Figure 3 presents the flow of trading, where the x -axis shows the trading days and the y -axis shows the portfolio value. Dashed line equates to passive trading strategy, while the solid line to MTS using the LSTM trading strategy. The greater the difference of solid line, compared to dashed line, the higher the excess profit. Passive trading strategy implements a single transaction and single transaction costs. From the day 450, passive strategy starts losing portfolio value below the initial amount of cash. In the end it scores 25.28% of loss. Although the LSTM on the other hand implements 6 transactions and thus increases transaction costs significantly, it behaves much more beneficially. When the MTS detects that the stock price is about to rise, it buys stocks, and when detects the price to fall, it sells them. The LSTM scores the 16.02% of profit and thus overcomes the passive trading strategy for more than 55%.

Table 2: Algorithm setup.

Parameter	Value
Batch size b	15
No. of epochs	1000
Learning rate	0.001
Optimizer	Adam
No. of units $LSTM_1$	10
No. of units $LSTM_2$	10

5. CONCLUSIONS

Efficient market hypothesis states that technical analysis is worthless for predicting the stock market performance. To test the validity of this hypothesis, we have applied the stacked long short-term memory network for trading the German stock Daimler AG from year 2010 to 2019. Back-testing approach was used to evaluate the quality of trading decisions. Implemented trading strategy significantly outperformed the passive trading strategy and found many arbitrage opportunities and other inefficiencies. Obtained results coincide with the results from a more detailed study in [4]. Results show that it is possible to generate higher-than-normal returns by relying on the technical analysis

²<https://keras.io>

Figure 3: Comparison between the automated MTS with LSTM trading strategy and passive trading strategy.

only. We conclude that either stock is inefficient, or the technical analysis is indeed a very efficient tool. We agree that arbitrage opportunities are partly conditional to the level of volatility of stock prices, since the higher the volatility, the higher the arbitrage opportunities. Volatility to its nature comes with constant reception of new information – the EMH states that stock prices quickly adjust to new information. If new information is constantly received, volatility is necessarily implied as well. For future work, we would like to implement a multi-stock MTS with LSTMs. Instead of single-stock returns, we would like to examine the portfolio returns. For example, German stocks in stock index DAX30 would be suitable for such an analysis.

6. REFERENCES

- [1] J. Brownlee. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Jason Brownlee, 2017.
- [2] Z. Ding, R. Xia, J. Yu, X. Li, and J. Yang. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer, 2018.
- [3] E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.
- [4] D. Fister, J. Mun, V. Jagrič, and T. Jagrič. Deep learning for stock market trading: A superior trading strategy? *Neural Network World*, 29(3):151–171, 2019.
- [5] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999.
- [6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [8] X.-H. Le, H. V. Ho, G. Lee, and S. Jung. Application of long short-term memory (lstm) neural network for flood forecasting. *Water*, 11(7):1387, 2019.
- [9] L. Lin, S. Gong, T. Li, and S. Peeta. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles. In *The Autonomous Vehicles Symposium*, volume 2018, 2018.
- [10] B. G. Malkiel. The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82, 2003.
- [11] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya. Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats. *Computers in biology and medicine*, 102:278–287, 2018.
- [12] L. H. Pedersen. *Efficiently inefficient: how smart money invests and market prices are determined*. Princeton University Press, 2015.
- [13] H. Shin and S. Y. Sohn. Segmentation of stock trading customers according to potential value. *Expert systems with applications*, 27(1):27–33, 2004.
- [14] V. Šonje, D. Alajbeg, and Z. Bubaš. Efficient market hypothesis: is the croatian stock market as (in) efficient as the us market. *Financial theory and practice*, 35(3):301–326, 2011.
- [15] R. Vinayakumar, K. Soman, P. Poornachandran, and S. Sachin Kumar. Detecting android malware using long short-term memory (lstm). *Journal of Intelligent & Fuzzy Systems*, 34(3):1277–1288, 2018.
- [16] B. W. Weber. Screen-based trading in futures markets: recent developments and research propositions. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, pages 10–pp. IEEE, 1999.
- [17] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275:167–179, 2018.