Acta Univ. Sapientiae Informatica **13,** 2 (2021) 288–302

DOI: 10.2478/ausi-2021-0013

Trading sparse, mean reverting portfolios using VAR(1) and LSTM prediction

Attila RÁCZ
Budapest University of Technology and Economics
Department of Networked Systems and Services
email: racz@hit.bme.hu

Norbert FOGARASI
Budapest University of Technology and Economics
Department of Networked Systems and Services
email: fogarasi@hit.bme.hu

Abstract. We investigated the predictability of mean reverting portfolios and the VAR(1) model in several aspects. First, we checked the dependency of the accuracy of VAR(1) model on different data types including the original data itself, the return of prices, the natural logarithm of stock and on the log return. Then we compared the accuracy of predictions of mean reverting portfolios coming from VAR(1) with different generative models such as VAR(1) and LSTM for both online and offline data. It was eventually shown that the LSTM predicts much better than the VAR(1) model. The conclusion is that the VAR(1) assumption works well in selecting the mean reverting portfolio, however, LSTM is a better choice for prediction. With the combined model a strategy with positive trading mean profit was successfully developed. We found that online LSTM outperforms all VAR(1) predictions and results in a positive expected profit when used in a simple trading algorithm.

1 Introduction

The usefulness of mean reverting portfolios was discussed several times before [2], [1], [3], [4]. Maximizing the predictability helps to create simple, effective

Computing Classification System 1998: G.1.6.
Mathematics Subject Classification 2010: 91B60, 91B84
Key words and phrases: portfolio selection, optimization, mean reversion, time-series prediction, LSTM

and robust trading strategies. Our previous work based on calibrating the VAR(1) generative model produced acceptable results on the S&P500 stock prices [10]. In the first part of this work we investigate how accurate the VAR(1) model is on different data models. Data models could mean the representation of the data, ie. the raw stock data itself or its transformation. Using a well performed data model we generate sparse portfolios with high predictability by searching the maximal generalized eigenvalue of the regression matrix. The second part of this work focuses to the prediction of the already created portfolio. The first strategy is to utilize the calibrated regression matrix of the VAR(1) model, the second is to train an LSTM neural network on the portfolio and use it for prediction [13], [11]. Both methods were tested for online and offline data. Prediction on online data means we use only the available real time data for future values while offline means we incorporate the previously predicted values to regress the next portfolio value. The latter makes it possible to predict accurately for a longer term. The structure of the paper is the following:

- In section 2, we briefly discuss the concept of VAR(1) model and mean reversion. Then we make a comparison of the accuracy of the VAR(1) model on the different data models.
- In section 3, we discuss the concept of LSTM neural network and how useful it is on mean reverting time series.
- In section 4, we discuss the details of the concept of online and offline prediction.
- In section 5, we compare the performance of the different techniques.
- In section 6, we make conclusions and recommendations.

2 VAR(1) model on derived time series

This section briefly explains the mean reverting processes and the modeling of the stock data with VAR(1). We call a stochastic process mean reverting when the value of the process oscillates around its average value. When the price is below its long-term mean it will likely increase rather than decrease and vice versa. This supports building a simple trading strategy and to estimate the trading range for the portfolio.

2.1 Ornstein-Uhlenbeck process

Mean reverting processes are formalized by the so-called Ornstein-Uhlenbeck process [8]. Let's denote the price of our portfolio by p_t at time t, and by s_t^i the price of the i^{th} stock at time t. Our mean reverting portfolio p_t is composed by the linear combination of s_t^i's. The stochastic differential equation that drives the Ornstein-Uhlenbeck process is

$$dp_t = \lambda(\mu - p_t)\,dt + \sigma dW_t \tag{1}$$

where W_t is a Wiener process, σ is a parameter proportional to the standard deviation of the Wiener process, λ is the speed of mean reversion and μ is the long-term mean of the process reverting to. The deterministic part of the stochastic differential equation (SDE) represents the property and that the magnitude of attraction to the long-term mean is proportional to the distance from the mean. The solution of stochastic differential equation is:

$$p(t) = p(0)\,e^{-\lambda t} + \mu\left(1 - e^{-\lambda t}\right) + \int_0^t \sigma e^{-\lambda(t-s)} dW(s) \tag{2}$$

The expected value of equation (2) is

$$\mathbb{E}[p(t)] = p(0)\,e^{-\lambda t} + \mu\left(1 - e^{-\lambda t}\right) \tag{3}$$

and the variance is

$$\mathbb{V}[p(t)] = \sigma^2 \int_0^t e^{-2\lambda(t-s)} ds = \sigma^2 \frac{1 - e^{-2\lambda t}}{2\lambda}. \tag{4}$$

Consequently in very long-term, the expectation converges to

$$\lim_{t\to\infty} \mathbb{E}[p(t)] = \mu, \tag{5}$$

while the variance is

$$\lim_{t\to\infty} \mathbb{V}[p(t)] = \frac{\sigma^2}{2\lambda}, \tag{6}$$

Note the variance is inversely proportional to the speed of mean reversion.

2.2 Asset dynamics and portfolio selection

2.2.1 Modeling asset dynamics with VAR(1)

In [2], [3] the concept of predictability was introduced in the following way:

$$\nu = \frac{\sigma_{t-1}^2}{\sigma_t^2} \tag{7}$$

where σ_t^2 is the variance of the time series. If the denominator is larger in (7), S_t will be pure noise as t goes to infinity therefore the time series is completely unpredictable, however while the nominator is larger as t going forward S_t will be perfectly predictable. Let the dynamics of the assets be modeled as discrete *vector autoregressive process* with parameter 1. Generally, VAR(p) means that the regression uses the last p values in the time series. As mentioned before $s_{i,t}$ denotes the price of the stock i at time t where $i = 1, \ldots, n$, where n is the size of the set of stocks. The most general model used in article 8 is the non-stationary VAR(1) model that contains a time independent constant scalar shift term to describe drift or ie. to ensure positivity of the elements for all t:

$$s_{t+1} = c + As_t + W_t, \tag{8}$$

where A is an n by n real matrix constant at some certain time period, c is a time independent real scalar constant, W_t represents the noise or error term of the model with zero mean value, some constant variance and uncorrelated across time. This can be rewritten in a concise VAR(1) notation by incorporating shift into the matrix of auto regression:

$$s'_{t+1} = A's'_t + W'_t \tag{9}$$

extending the notations

$$\begin{bmatrix} s'^1_{t+1} \\ \vdots \\ s'^n_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} & c_1 \\ \vdots & \ddots & \vdots & \\ a_{n,1} & \cdots & a_{n,n} & c_n \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} s'^1_t \\ \vdots \\ s'^n_t \\ 1 \end{bmatrix} + \begin{bmatrix} W^1_t \\ \vdots \\ W^n_t \\ 0 \end{bmatrix} \tag{10}$$

where A' refers to a $(n+1) \times (n+1)$ matrix in which the last column is filled with the constant shift c, the last row has *zeros* except the element of $(n+1)^{st}$ which should be strictly 1, x'_t a vector with $n+1$ elements strictly 1 at the $(n+1)^{st}$ element and W'_t still provides the noise as in the previous case except no noise for the $(n+1)^{st}$ element [7]. Henceforth, we ignore the *prime* sign in this article. The auto regression matrix in equation (8) can be approximated using least squares regression by

$$\hat{A} = \left(s^T_{t-1} s_{t-1}\right)^{-1} s^T_{t-1} s_t \tag{11}$$

Using this model a portfolio can be created with a linear combinations of the assets. To make this, let P be a real valued vector. This vector represents the

weights that relate to stocks. The time evolution of the value of our portfolio can be written as

$$P s_t = PAs_{t-1} + PW_t \tag{12}$$

After using the definition of predictability (7) for the $VAR(1)$ model we get:

$$\nu(P) = \frac{var\left(P^T A s_{t-1}\right)}{var\left(P^T s_{t-1}\right)} = \frac{E\left(P^T A s_{t-1} s_{t-1}^T A^T P\right)}{E\left(P^T s_{t-1} s_{t-1}^T P\right)} \tag{13}$$

As only s_t is stochastic, A and P can be factored out from the expectation calculation. So eventually we have the covariance matrix of the time series, which we denote by G. Maximizing predictability is eventually a generalized eigenvalue problem:

$$P_{opt} = argmax\left(\nu(P)\right) = argmax\left(\frac{P^T A G A^T P}{P^T G P}\right), \tag{14}$$

The argument of the $argmax$ operator is the so called Rayleigh quotient, where the above becomes the following:

$$AGA^T P = \lambda \tag{15}$$

Current scenario is to keep the number of constituents low, which is an additional constraint to the optimization. On the other hand, to hold the transaction cost as low as possible and also to keep the portfolio complexity low, only a low number of stocks will need to be enabled. The optimization problem now is the trade-off between the maximization of mean reversion speed and the minimization of the cardinality of stocks. Mathematically the equation (14) has an additional constraint

$$\begin{aligned} &P_{opt} = argmax\left(\nu(P)\right) = argmax\left(\frac{P^T A G A^T P}{P^T G P}\right), \\ &\text{subject to } \mathbf{Card}(P) \leq k \end{aligned} \tag{16}$$

The above optimization was performed in two steps. First a suboptimal solution is found by a greedy algorithm. In the next step Simulated Annealing (SA) was applied, as in [12], [5] and [4]. The starting point of the SA was the solution of the first step. To model the dynamics of the stock prices $VAR(1)$ model with or without constant shift being used. However the calibrated regression matrix not applied for predict individual stock prices only to find it's maximal generalized eigenvalue, it is worth to investigate how accurate the

	Without constant shift	With constant shift
Normal data	0.0047619	0.08424908
Normal return	0.07875458	0.10769231
Log Data	0.0021978	0.23736264
Log return	0.19157509	0.29340659

Table 1: Relative frequency of the highest accuracy

regression model for different data types. Here, we will now discuss whether it is useful to apply the calibration on derived data such as the return, the natural logarithm or the logarithmic return of stock prices. The tests were performed on S&P500 data with many configurations. A point in this configuration space includes the width of the calibration window and the starting point. The particular time window range varied between 50 days to 400 days, the end points moved between $2016-01-01$ and $2021-01-01$. The calibration performed with and without the time independent shift for every data model. To be able to compare the accuracy, every regressed data other than normal data (i.e. normal return, log and log return) were transformed back to normal. Then we calculated the mean squared error and recorded which data model has the lowest error for a certain configuration. The results in relative frequency are summarized in Table 1. In this table we can see that higher accuracy is reachable when we incorporate the constant shift into the VAR(1) model. That also applies to a large extent of the cases the log or log return data model are the most accurate. In the future we use the log model to test and compare the effectiveness of different trading algorithms.

An example of regression with constant shift is in Figure 1. Here all the four prediction of data model (ie. the original data, simple return(diff), natural logarithm of the original and the log return) and values of the asset (Adobe) are represented. Here the regressed values can vary to a large extent compared to the real stock values. As mentioned previously the aim is not to predict individual stock prices but create a portfolio which has the highest predictability.

3 Predicting mean-reverting portfolio with LSTM

3.1 LSTM introduction

Long Short Term Memory networks (LSTM) are recurrent neural networks (RNN), designed to learn long-term dependencies by [6]. As an RNN, LSTMs have a repeating modules and each module has the similar structure with four,

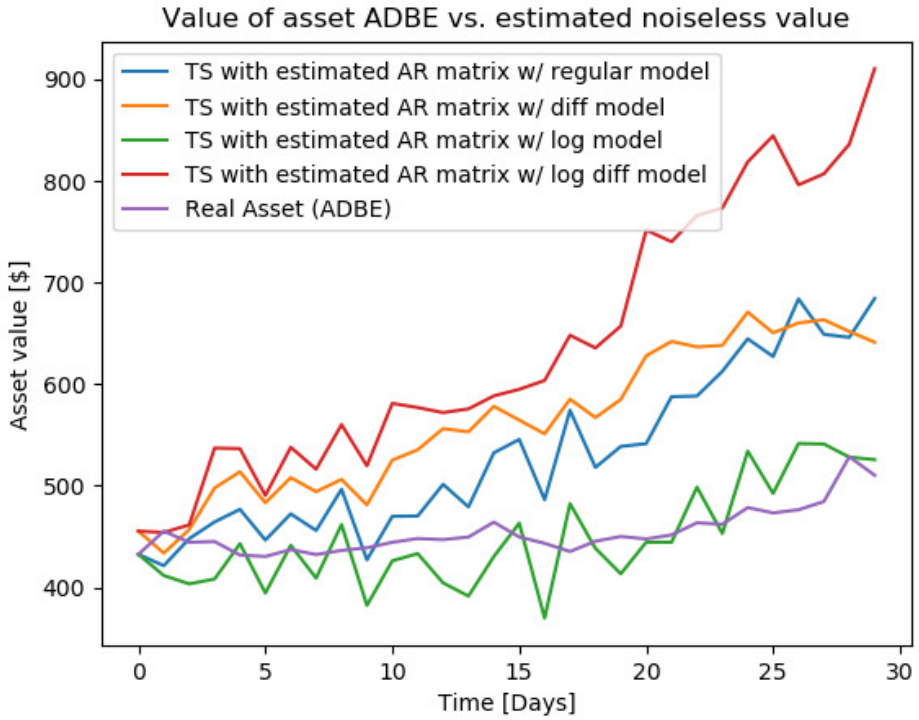

Figure 1: Result of prediction for every data model with constant shift on an example stock (Adobe).

interacting layers. The most important constituent for long-term learning is the state of the module or cell, which is represented by the top horizontal line. It interacts with other layers in the cell via pointwise multiplication ans addition and provides input for the next cell. The way how an LSTM cell constructed by gates makes it possible to remove or add information to the cell state. Since the sigmoid function outputs numbers between zero and one, it can provide weights to each components. A cell in an LSTM network contains three of these sigmoid gates. The first gate is the forget gate which is to decide what information have to be thrown away from the cell state. It takes the h_{t-1} hidden state from previous cell and x_t input, and the output multiplicated with the cell state C_{t-1}.

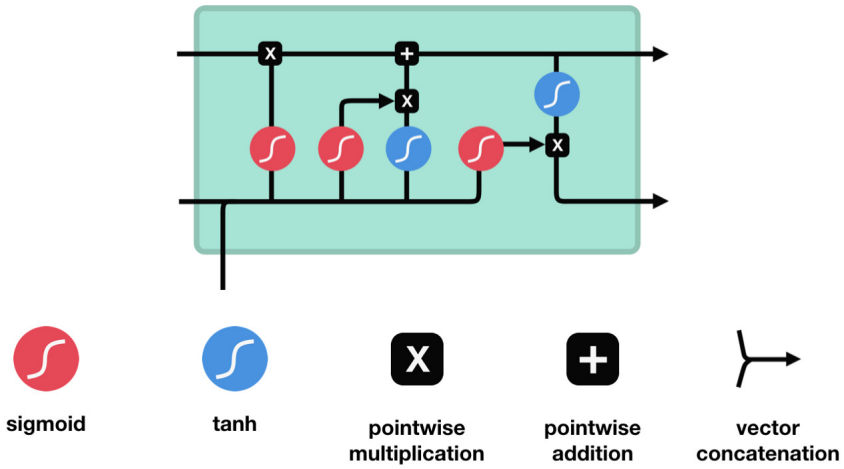

Figure 2: Schematic diagram of LSTM network. Source: [9]

$$o_{forget} = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{17}$$

where σ is the `sigmoid` function, W_f and b_f represents the network in forget gate. The following gate, the `input gate` is for to decide what information should be updated in the cell state which is consist of two layers with two different activation function.

$$f_{input} = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{18}$$

where W_i and b_i represent the neural network of the layer. The output of the input gate is multiplicated with the output of the `tanh` layer that creates a vector which will be used to update cell state values, C_t by pointwise adding the multiplicated result to the old cell state vector C_{t-1}.

$$f_C = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{19}$$

The final gate, the `output gate` decides what should hidden state values should be transferred to the next cell. The output will be the combination of the previous hidden state values, inputs of the actual cell and the updated cell state values.

$$C_t = o_{forget} * C_{t-1} + f_{input} * f_C \tag{20}$$

First the third sigmoid layer decides what parts of the cell state will be treated as output.

$$o_{output} = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{21}$$

The cell state C_t runs through the tanh layer, this makes the state values between -1 and 1, and multiply it by the output of the sigmoid gate o_{output}.

$$h_t = o_{output} \tanh(C_t) \tag{22}$$

.

The state variables h_t and C_t serve as input for the next module.

3.2 Predicting

The VAR(1) model was applied to fit an analytical model to the data model and from that a well predictable portfolio was created. The regression matrix can be used to predict the future values of the portfolio. However, the prediction is not based on the portfolio itself but on the constituents. Hence, the errors accumulated during the calibration against each stock make the predicted time series very noisy and inaccurate. It is much more reasonable to use the historical values of the portfolio itself to regress future prices. We constructed and trained LSTM recurrent neural network to predict future prices of our portfolio. We trained the network on the same time range as used for VAR(1) calibration. So we created the portfolio with the regression matrix and used as an input for the training. The number of LSTM layers is 4, batch size is 1, epochs is 100. We used 3 consecutive data to predict. In Figure 3 an example of the testing can be seen.

4 Online vs Offline prediction

The common property of the two methods, which are VAR(1) and LSTM is that they are only able regress one time step ahead. The online prediction is fully based on real data. As both VAR(1) and simple LSTM predict only t time step ahead, to regress a time interval we should use previously predicted data, as well.

Online prediction: Using only real data for prediction the next value can only be estimated when all real data are available.

Offline prediction: Using both real and predicted data, longer terms can be predicted by utilizing the previously estimated data as input for the regression.

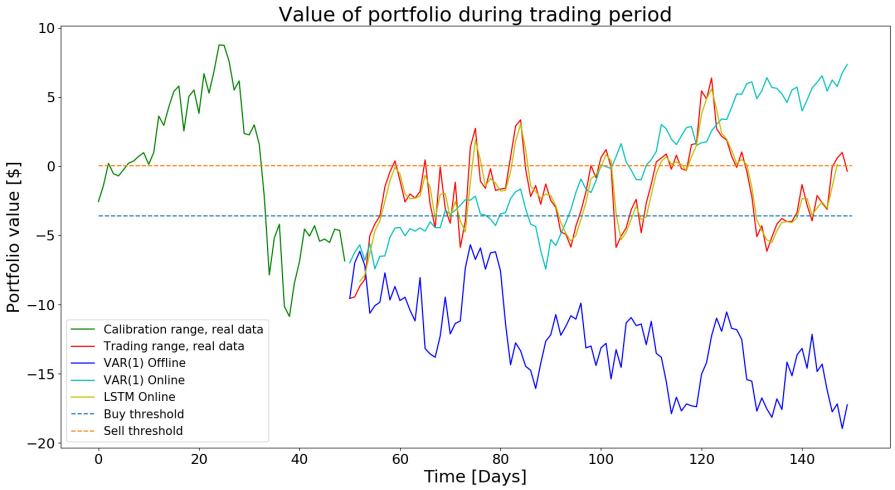

Figure 3: Predicting optimized portfolio. Green curve is the calibration range, blue VAR(1) prediction, red curve real value, yellow is the LSTM prediction.

4.1 Trading strategies

We can build trading strategies by combining the online and offline predictions with the regression methods. The definition of the trading range is essential. It can happen retrospectively or prospectively. As we incorporate the LSTM or VAR(1) regression values into the simple mean-reverting trading strategy the information they carry modifies the logic. As the selling or buying events are triggered by the value of the portfolio, the final decision is affected by the value of the prediction. As if the trader has cash in hand and the purchase event is triggered the real buying will be performed if the next estimated value is higher than the current. Otherwise at least one step will be waited out. Similar logic occurs when there is a portfolio in hand, see Figure 4.

Online strategies: The online strategy uses only the online prediction. Therefore the trading range can be estimated using the historical data.

Offline strategies: The offline trading strategy involves long-term price prediction regressing still only one time step ahead and incorporating predicted data for farther estimation. The method simply involves more predicted data successively as regression goes further ahead. This provides another method to

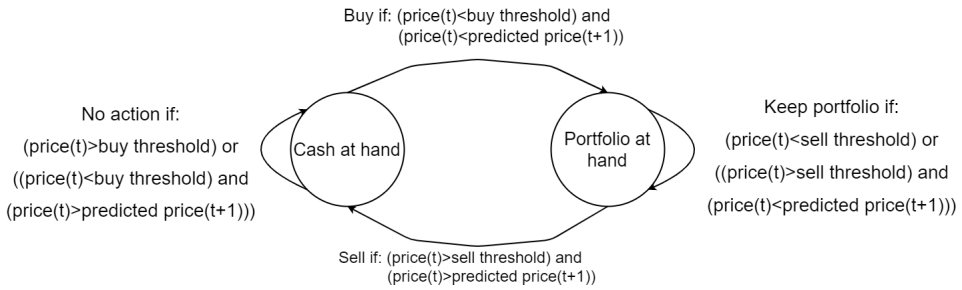

Figure 4: Schematic diagram of the general trading workflow.

estimate the trading range, which is utilized in this strategy both for VAR(1) and LSTM.

5 Performance test

The performance tests were carried out for all the 4 trading configurations, ie. combining online and offline prediction with VAR(1) and LSTM methods. The initial invested amount was \$10000. Each configuration was run with 5 different sparsities: $(3, 5, 8, 11, 14)$. The data was S&P500 daily close stock prices between 01/01/2016 and 12/31/2020. For a sparsity the trading was repeated 30 times with different calibration window lengths and positions.

The length of trading time was constrained to 100 days. This was necessary to be able to compare each trading cases, as we have limited length of data and the possible trading length could vary from few weeks to few years. In case when portfolio was at hand at the end of trading range because price did not hit sell threshold, the return was calculated by the actual value of portfolio subtracted the price at buying. During the test we see that at most 2 buy-sell events were performed. When 0 events was performed it was mostly due the trading range was not estimated precisely and the buy threshold was not hit. At this point we did not incorporate the trading fees as our work focused on the comparison of trading strategies detailed above.

In Table 2 we can see the mean gain and the standard deviation of returns for the used strategies.

On figures (5), (6) the distribution of the profits for the used strategies are visualized. We tested the strategies against different trading starting points and calibration window length $(50, 100, 150, 200)$. We applied a floor and ceiling functions for below $-\$100$ and above \$100 respectively only to make the

	3	5	8	11	13
Offline VAR	-3.31/39.52	2.38/20.04	0.30/25.42	1.03/18.53	-0.72/19.49
Offline LSTM	-2.88/30.52	1.08/23.18	-1.37/15.96	1.91/22.62	-0.69/19.38
Online VAR	4.73/28.15	7.86/40.79	3.97/27.32	4.97/31.06	5.88/35.79
Online LSTM	10.51/27.19	8.21/23.16	10.45/20.24	7.41/23.81	9.37/20.09

Table 2: The mean and standard deviation in terms of [$] of trade performance for trade methods and sparsity

useful part of distribution more detailed, this did not affected the calculation of the mean and standard deviation in Table 2.

It is clear from the table and from the figures that the most effective strategy is the online LSTM. It was able to consistently produce positive profits. Note that the offline data plots on Figures 5 and 6 very high peaks can be seen at 0 profit. These are due to inaccurate estimation of the trading range and no action was performed.

6 Conclusion and future works

We have presented how accurately we predict the S&P500 stocks with VAR(q) using different data models. We conclude that it is worth calibrating with log or log return data model. In the second part we have explained how the prediction of the sparse mean reverting portfolio, which was created by VAR(1), can be improved. The tests were performed on online and offline prediction. We modified the simple mean reverting trading logic by adding the predicted values to the trading decision process. We found that online LSTM outperforms all VAR(1) predictions and results in a positive expected profit when that is used in a simple trading algorithm in case of online prediction. One of the crucial things to increase profit is to estimate the trading range very precisely. One possible way is to create a Seq2seq LSTM neural network, that is able to predict more than one time step ahead. This can be the direction of future research.

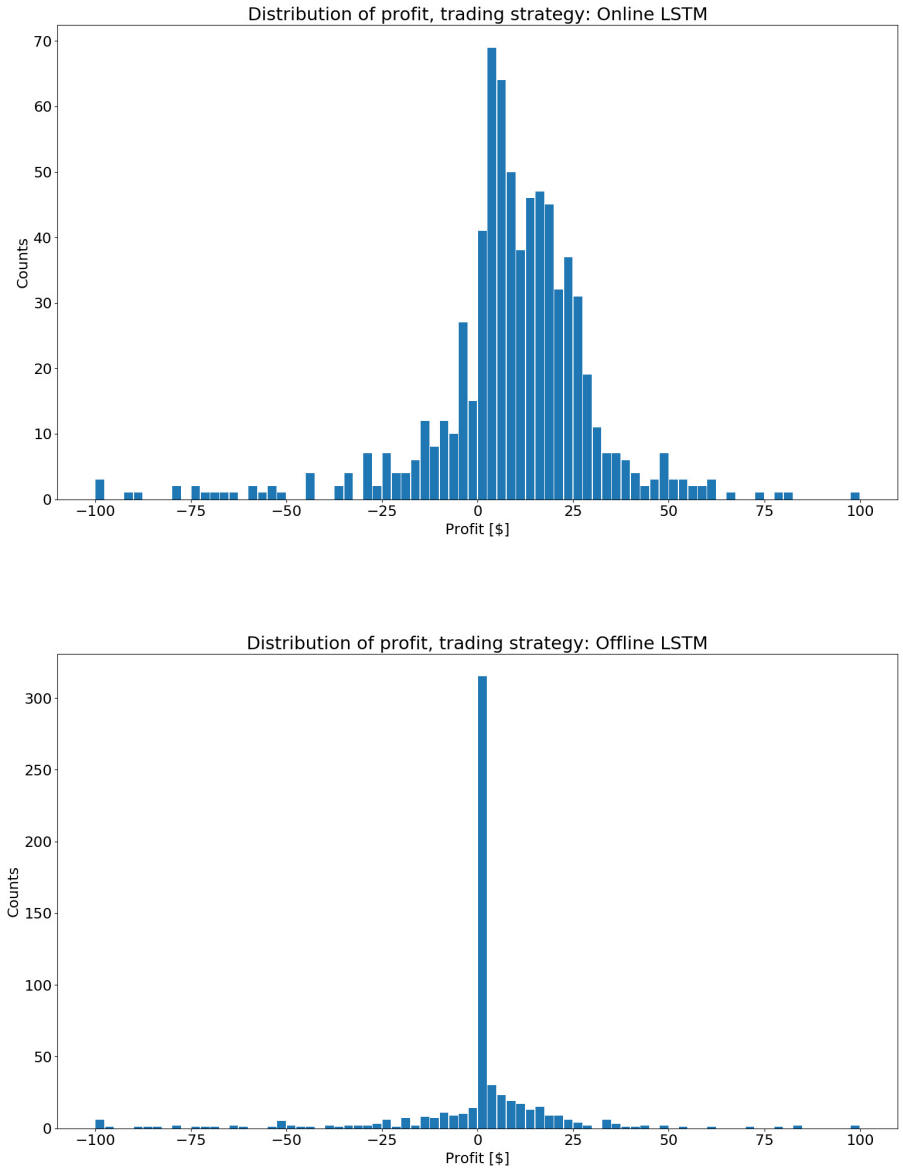

Figure 5: Profit histogram of LSTM prediction on online and offline data.

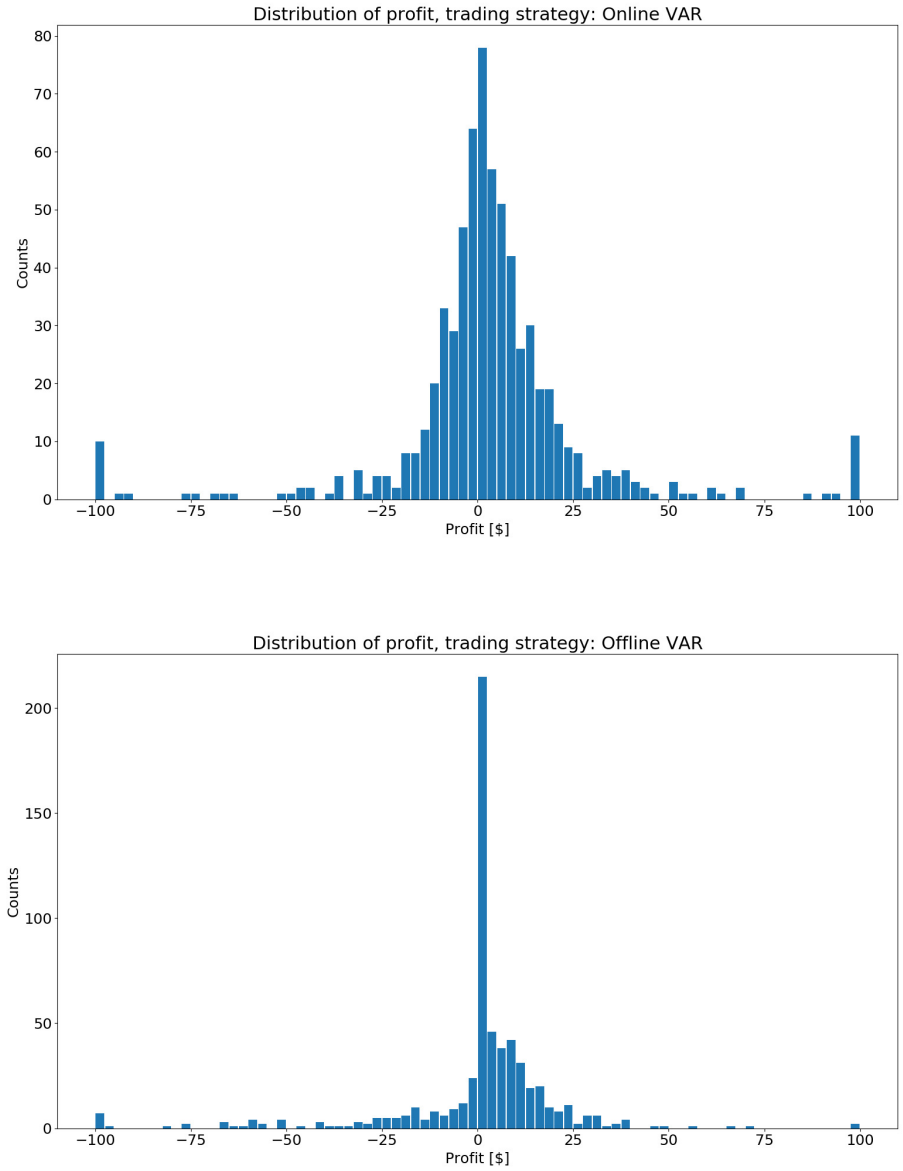

Figure 6: Profit histogram of VAR prediction on online and offline data.

References

[1] O. Banerjee, L. El Ghaoui, A. d'Aspremont, Model selection through sparse maximum likelihood estimation,*J. Mach. Learn. Res.*, **9** (2008) 485—516 ⇒288

[2] G. E. Box, G. C. Tiao, A canonical analysis of multiple time series *Biometrika*, **64** (1977) 355 ⇒288, 290

[3] A. d'Aspremont, Identifying small mean reverting portfolios, *Quant. Finance*, **11** (2011) 351–364 ⇒288, 290

[4] N. Fogarasi, J. Levendovszky, Sparse, mean reverting portfolio selection using simulated annealing,*Quant. Finance*, **11** (2011) 351–364 ⇒288, 292

[5] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Trans. Patt. Anal. Mach. Intellig*, **6** (1984) 721–741 ⇒292

[6] S. Hochreiter, J. Schmidhube, Long Short-Term Memories, *Neural Computation*, **9** (1997) 1735–1780 ⇒293

[7] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*, Springer (1993) ⇒291

[8] L. S. Ornstein, G. E. Uhlenbeck, On the theory of the Brownian motion, *Phys. Rev*, **36** (1930) 823 ⇒290

[9] M. Phi, Illustrated Guide to LSTM's and GRU's: A step by step explanation, Sep 24, 2018. https://tinyurl.com/aus2xtjx ⇒295

[10] A. Rácz, N. Fogarasi, Improved sparse mean reverting portfolio selection using Simulated Annealing and Extreme Learning Machine, submitted to *Algorithmic Finance* ⇒289

[11] V.-D. Ta, C.-M. Liu, D. A. Tadesse, Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading, *Applied Sciences*, **10** (2020) 437 ⇒289

[12] P. Salamon, P. Sibani, R. *Frost, facts, conjectures, and improvements for simulated annealing,* SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics, **36** (2002) ⇒292

[13] A. Yadav, C. K. Jha, A. Sharan, Optimizing LSTM for time series prediction in Indian stock market, *Procedia Computer Science*, **167** (2020) 2091–2100 ⇒289

Received: October 6, 2021 • Revised: November 3, 2021