Time Series Prediction using LSTM and Linear Regression

Dasten Turlin, Yerzhan Tolysbek and Nursultan Altayev

[1] *Nazarbayev Universty, School of Sciences and Humanities, Nur-Sultan*
[2] *Nazarbayev Universty, School of Engineering and Digital Sciences, Nur-Sultan*
[3] *Nazarbayev Universty, School of Engineering and Digital Sciences, Nur-Sultan*
[4] *Link to the corresponding video: https://youtu.be/cwvVuuxhxKE*

May 6, 2020

Abstract - Which is the best way to forecast time series? Effectively forecasting the failure data in the usage stage is essential to reasonably make reliability plans and carry out reliability maintaining activities. Beginning with the historical failure data of complex system, a long short-term memory (LSTM) based recurrent neural network for failure time series prediction is presented, in which the design of network structure, the procedures and algorithms of network training and forecasting are involved. Our LSTM model is a simple LSTM model, which is a current state-of-the art in financial time series prediction. Here, we managed to make the mean average error (MAE) as low as 9.325 (with 2000 epochs), which is a great result and we can claim that our LSTM can be a benchmark (top RNN networks achieve MAE around 5). However, our goal of this project is to obtain whether Multiple Linear Regression model can outperform or at least match current LSTM neural networks. After several attempts, the best result we got on the test set was MAE of 33.39, which, given seeming simplicity of decreasing the error by including more explanatory variables, tells us that it is arguably possible to develop a robust linear regression model for time series forecasting. However, some assumptions made when implementing this model can be a problem in the future.

1 Introduction

Recurrent neural networks are well suited to supervised learning problems where the dataset has a sequential nature. Time series forecasting should not be an exception. RNNs are essentially neural networks with memory. They can remember things from the past, which is obviously useful for predicting time-dependent targets. Yet, applying them to time series forecasting is not a trivial task. However, it is possible that older techniques like linear regression are capable of doing it too. This project aims to find high-performing ways to predict the values of Standard and Poor 500 (SP500) index.

2 Methodology

RNNs are neural networks for sequential data — hereby we apply them to time series. The main idea behind recurrent neural networks is using not only the in-put data, but also the previous outputs for making the current prediction. This idea makes a lot sense — we could build neural networks passing values forward in time. However, such simple solutions usually do not work as expected. They are hard to train and they are forgetful. Rather, we need to have a system with some kind of memory. Thereby, we use long short-term memory networks as benchmark algorithm.

2.1 Software

Our language of choice was Python 3 because of its simplicity and availability of data science packages we

heavily relied on. Speaking about them, throughout out project we used PyTorch for LSTM setup and training, and Numpy and Pandas for data processing.

2.2 Dataset

We retrieved SP 500 data from Kaggle and took company-wise data from Yahoo Finance. Both datasets were free to use (links are attached in the References section). It is a rarity when such datasets are available freely because usually they cost amounts of money. Both datasets contained information on market close prices of SP500 index and all 500 companies comprising this index, too. Time period was from 2013-02-08 to 2018-02-07, which is 1259 days in total. Then we divided this SP dataset into training an test sets, with the former containing 1000 entries and the latter containing 259 entries for the last 259 days.

2.3 LSTM

Long short-term memory (Hochreiter and Schmidhuber [1997]) is a gated memory unit for neural networks. It has 3 gates that manage the contents of the memory. These gates are simple logistic functions of weighted sums, where the weights might be learnt by backpropagation. It means that, even though it seems a bit complicated, the LSTM perfectly fits into the neural network and its training process. It can learn what it needs to learn, remember what it needs to remember, and recall what it needs to recall, without any special training or optimization. The input gate and the forget gate manage the cellstate , which is the long-term memory. The output gate produces the output vector or hidden state, which is the memory focused for use. This memory system enables the network to remember for a long time, which was badly missing from vanilla recurrent neural networks. Indeed, LSTM is a better version of a simple RNN because it effectively solves vanishing and exploding gradient problem, when gradient updates either converged to zero or infinity.

2.4 Linear Regression

Linear regression is the one of the most popular statistical technique commonly used for predictive modeling. Breaking it down to basics, it comes to providing a linear combination of independent variables, on which our target variable is built upon:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \varepsilon_i,$$

where every obtained coefficient is

$$\beta_i = Cov(x_i, y)/Var(x_i)$$

Using already existing function of Linear Regression from Scikit-learn, we experimented with regression on different parts of our graph to finally expand it.

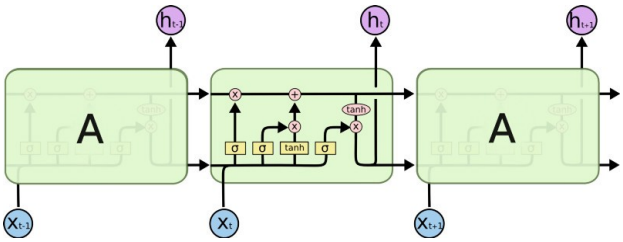
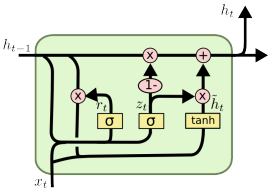


Figure 1: *LSTM architecture*



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2: *LSTM formulas*

2.5 Training process

Our LSTM model had 1 hidden layer with 32 neurons, and a varying number of training iterations from 100 to 2000. When training LSTM, we divided our training set into many subsequences of different length, usually 8. Then, we fed the model these sequences to obtain the gradients using backpropagation. Then, after backpropagation and ADAM optimization, we used these sequences to generate new values and compare them with the actual values of SP500 at a given test period. In all cases, training error converged well.

However, linear regression model training contained one very important step related to parameters selection. To choose the dependent variable out of 500 stocks, we took the ones which had highest positive correlation with the SP500 throughout the whole training period, i.e. 1000 days. These 6 (later 8) stocks were chosen as the parameters for our model. Then, we calculated the coefficients using scikit-learn.

3 Results

These efforts have brought us some remarkable results. We have used 2 different models for training our data set. First one was LSTM using 2000 epochs. Its train error can be seen at Figure 3. As epochs increased train error decreased which is what we expected. Next is the graph of actual SP data and LSTM predicted data on Figure 4. Results of LSTM were very approximate to actual ones with absolute error equal to 9.017. In linear regression, firstly we ended up with results as in Figure 5. Trend was close to actual one however Y-values differed by a mild amount. Therefore we have updated our linear regression algorithm by adding 2 more companies to our training data and made same trainings. Updated results are more promising as they are more close to actual SP500 values as can be seen at Figure 6. That is very important because it shows that increasing the number of independent variables

Figure 3: *LSTM training error, epochs - 2000*

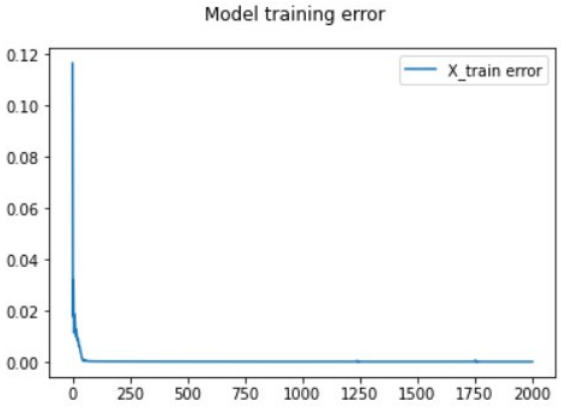


Figure 4: *LSTM with epochs = 2000*

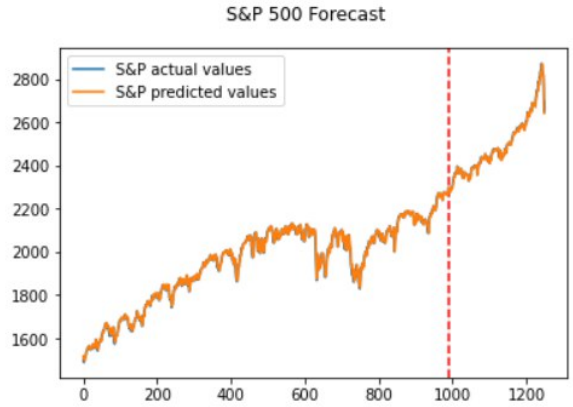


Figure 5: *Lin.regression with 6 independent variables*

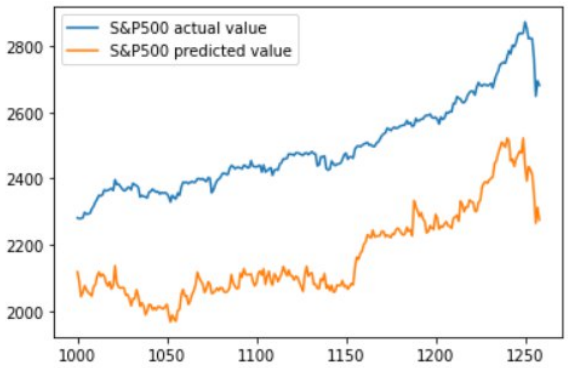
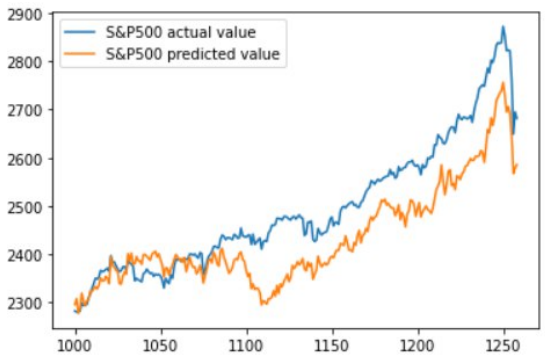


Figure 6: *Lin.regression with 8 independent variables*



decreases test error (although R-squared might fall)

LSTM Training error (MSE)

Epochs	LSTM training error (MSE)
50	0.00526
150	0.00025
250	0.00024
500	0.00017
2000	0.00013

LSTM versus Lin.Regression test error (MAE)

Epochs/Train set size	LSTM test error	Lin. Reg. test error
50	279.79	149.75
150	45.88	26.87
250	44.104	29.87
500	11.63	146.62
2000	10.51	N/A

4 Conclusion

In our study, our LSTM model is a simple LSTM model, which is a current state-of-the art in financial time series prediction. Here, we managed to make the MAE as low as 9.325 (with 2000 epochs), which is a great result and we can claim that our LSTM can be a benchmark (top RNN networks achieve MAE around 5).

Linear regression is relatively easy to use, and it has an easily-manageable error. However, despite the great results shown by it, the assumptions I made about the movement of SP index can be untrue. Correlation doesn't imply causation, and obviously there are many more factors influencing SP index, so our regression almost certainly lacks some more factors/features.

However, we still should confess that the performance of multiple linear regression was surprisingly good. Taking our LSTM model here as a benchmark, we haven't managed to get such low errors, but we showed viability of our method of choosing the stocks with maximum correlation with the original SP index. Moreover, we showed that increasing the number of explanatory variables does lower the error. However, overfitting is the possible issue here.

5 References

Wang, X. Wu, J. Liu, C. Yang, H. Du, Y. Niu, W.. (2018). Exploring LSTM based recurrent neural network for failure time series prediction. Beijing Hangkong Hangtian Daxue Xuebao/Journal of Beijing University of Aeronautics and

Cao, J., Li, Z., Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. Physica A: Statistical Mechanics and its Applications, 519, 127-139.

Cowles A. Can Stock Market Forecasters Forecast? Econom. Astronautics.

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Datasets:
https://www.kaggle.com/camnugent/sandp500
https://finance.yahoo.com/quote/

6 Contributions

Each student has contributed to the project equally by implementing each of the parts together.

However, Yerzhan and Nursultan has contributed more to the report of LSTM, and Dasten has contributed more to the last regression idea and implementation.