Stock market forecasting with temporal convolutional network

September 13, 2021

Lorenzo Di Fruscia

Abstract

Financial time series modelling is a research area of great interest since 1970. According to the Efficient Market Hypothesis, market should be unpredictable; nevertheless a lot of work has been done in the attempt to identify patterns in stock movement, with the goal of modelling non linear and non stationary stock prices. In this report i implement a temporal convolutional network (TCN) and use it to predict "open" stock prices of Google, having at disposal only historical records as informative prior. A simple long short-term memory (LSTM) model is also implemented in order to compare both performances on unseen data.

1. Introduction

Stock market is highly unpredictable due to the fact that stocks fluctuations depend on an enormous amount of variables, including local and global economic conditions, political conditions, natural disasters and so on. The Efficient Market Hypothesis EMH, which is still matter of debate to this day and is a cornerstone of modern financial theory, states that stock prices already reflect all the existent relevant information so that it is not possible to constantly beat the overall market through forecasting and market timing. Valid or not, the fact that financial time series are non-stationary, nonlinear and high-noise makes accurate prediction of stock prices (and not only) a really challenging task. From 1970 to this day, several approaches have been tried and recently deep learning algorithms came on the scene and started stealing the spotlight. On the subject, extensive research has been done using long short-term memory (LSTM) models, in order to predict the future evolution of stocks. In this report i implement a variation of convolutional neural network (CNN) called temporal convolutional network (TCN). The aim is to verify if a simple implementation of a TCN is capable of outperform a simple LSTM in terms of accuracy in stock forecasting.

Email: Lorenzo Di Fruscia <difruscia.1765562@studenti.uniroma1.it>.

Deep Learning and Applied AI 2021, Sapienza University of Rome, 2nd semester a.y. 2020/2021.

2. Related work

There are different interesting related works which implement LSTM to make stock price forecasting. One of them is (Mehtab S., 2020). In this paper the authors implement eight machine learning models and four different LSTM models, using univariate and multivariate historical data, with one or two weeks prior. The results are that LSTM models obtain far superior results in their capability of extracting features with respect to machine learning models. Furthermore, univariate analysis in LSTM models led to better results of multivariate one. One paper which inspired this report is (Shaojie Bai et al., 2018) the original paper on the implementation of TCN. Here the authors found that a simple convolutional architecture outperformed LSTM models in different tasks, demonstrating longer effective memory.

3. Dataset and Methods

The dataset consists of 10 years of Google 'Open' prices: 2516 days from 2011-01-01 to 2021-01-01 split in 80% training and 20% test. After that, data goes under first order differencing: returns are calculated from prices in order to remove non-stationary behaviour and ease training for the models; then, they are standardized with zero mean and unitary standard deviation. For both models, validation is done with an adapted version of cross validation for time series to take care of the causal nature of data: nested cross validation. Here number of splits is fixed to 4 so that validation set sizes are comparable with test set size. In the end, data is converted back from predicted standardized returns to predicted open prices for both models. Final training and test losses of both models are then compared with an intuitive baseline model: the "naive forecast" or persistent algorithm, which states that "the best prediction for tomorrow is today's value". Last but not least, up-down accuracy of both models is calculated and compared with a random walk baseline, where random guessing corresponds to 50% accuracy on future predictions.

Both models are first trained according to nested cross validation; their validation losses are saved and mean validation loss is computed as a function of epochs. Number of best epoch is then saved and used as fixed hyperparameter for the final training on the training dataset.

LSTM

The LSTM model i implemented makes daily forecast on returns with prior one week of historical data. It is a two stacked layer LSTM with dropout after the first layer, and a FC layer at the end.

TCN

The model works by taking advantage of dilated convolutions in order to increase the receptive field of the kernels as you go deeper with layers. Convolution operation needs to be causal and it is accounted by using asymmetric padding. It is made up of one block of four dilated 1D convolutions with weight normalization and dropout at every layer, and a FC layer at the end. The kernel size i used is 5 and the model has sufficient receptive field to look over more than 30 days back in the past in the final layer, and does daily forecast as for the LSTM. The TCN that i implemented here is a different and simplified version of what is proposed on the original paper for the tasks they tried to solve.

4. Results

Both models have been trained with mean squared error (MSE) as loss, with a batch size of 16 and a learning rate of 5×10^{-4}. Best results on open price returns out of five runs, for both models, are reported in table 1

Table 1. Validation and test losses on returns

Model	Model size	best epoch	Val. loss	Test loss
LSTM	$\simeq$ 3k	84	0.203	**0.317**
TCN	$\simeq$ 350	90	**0.063**	0.330

After that, returns have been reconverted in open prices and compared with naive forecast as a baseline for the loss. As expected, naive forecast led to a far smaller loss on test open prices. Results are in table 2

Table 2. Performance comparison pt.1

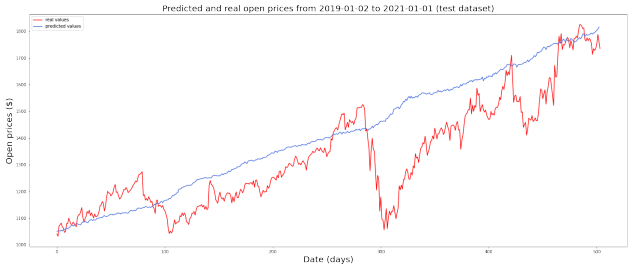
Results	LSTM	TCN	Naive
Test loss	22732	17736	**654**

In the end both models have been tested to calculate up-down accuracy with respect to a random walk behaviour, which has a model accuracy of 50%. Results using previous best run are reported in table 3.

Table 3. Performance comparison pt.2

Results	LSTM	TCN	RW
Accuracy	**52.5%**	48.7%	50%

Plots of test predictions on open prices respectively for LSTM and TCN are shown in figure 1 and figure 2:



5. Conclusion

In this report implemented a simple TCN model and confronted it with a simple LSTM on a stock forecasting task. Despite the lower validation loss on TCN, LSTM has a slightly better test loss and performs better in terms of up-down accuracy. However TCN is capable of obtaining similar results to LSTM with fewer parameters. Their results are similar on test set, following the mean behaviour rather than modeling the real data. One way to improve general results could be with the implementation of additional information to put beside historical data. A possible modification on the way training has been done could be implementing a rolling window cross validation (forward chaining) instead of the extended window method here presented and see how the models respond to it. One possible future implementation for TCN includes a deeper network with a wider receptive field; architecture could also be changed to host a self attention layer.

The code associated with the report here presented can be found at: `https://github.com/lorentzDFR/DLAI_project.git`

References

Mehtab S., Jaydip S., A. D. Stock price prediction using machine learning and lstm-based deep learning models. *SoMMA'20*, 2020.

Shaojie Bai et al., J. Zico Kolter, V. K. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. 2018.