Stock Price Forecasting using GRU

June 20, 2021

Leonardo Emili Alessio Luciani

1. Introduction

Stock price prediction is an active field of study where the objective goal is to predict the trend of the stock market, typically based on its historical evidence. Despite the theory of *efficient markets* claims that there cannot be any pattern in the trend of financial assets due to the overall knowledge of the mass, some affirm that periods of human behavioral irrationality lead to strong operation correlation reflected in the markets (Lo, 2017). Therefore, the task of predicting stock prices based on past data seems not to be completely inapproachable. Over the years, many methods have been proposed to tackle the task: ranging from the Naive Forecast approach, which trivially forecasts the stock value to be the last observed one, up to the most recent ones that heavily rely on machine learning techniques. In this context, we will delve into the topic of stock price prediction using fundamental data to assess whether a stock is attractive to investors. This technique, as opposed to technical analysis, considers prices, as well as financial reports, to model the problem. Here, the underlying assumption is that one cannot tell whether it is worth buying a stock only by looking at its current price and volumes. Indeed, quarterly released financial reports about companies and external information (e.g. the common sentiment on a stock ticker) may be crucial to assess the quality of a given stock. In this project, we experimented with the effectiveness of deep learning techniques applied to the stock price prediction task. The key idea is the use of recurrent neural networks that are able to exploit temporal dependencies of events, hence conditioning the presence of an event on previous events.

2. Related work

Some approaches have been explored in this direction, such as in (Zou & Qu, 2020) where the authors apply LSTM, Stacked-LSTM, and Attention-Based LSTM into the prediction of stock prices. The authors also propose an evaluating framework to assess the quality of their models based on the return of the trading strategy. In another work (Mehtab et al., 2020), the authors predict the open value of NIFTY 50 using different machine learning and deep learning models. They also demonstrate that using one-week prior data as input leads to good results.

Email: Leonardo Emili <emili.1802989@studenti.uniroma1.it>, Alessio Luciani <luciani.1797637@studenti.uniroma1.it>.

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

3. Dataset

Data wrangling operations have been crucial for this task. In fact, we start considering the S&P 500 stock data, which is a collection of daily stock prices for all companies from the S&P 500 index. This dataset provides us with the following features: open price, high price, low price, close price, volume, stock ticker, and date. We also consider an auxiliary dataset that provides us with fundamental data from Yahoo Finance. The dataset contains the following columns: Forward P/E, DE Ratio, Earnings Growth, Enterprise Value/EBITDA, EBITDA, Current Ratio, Cash Flow, Trailing P/E, Beta, PEG Ratio, Gross Profit, Total Debt, Price, Return on Equity, Return on Assets, Price/Book, Revenue Growth, Operating Margin, Enterprise Value/Revenue, Revenue, Total Cash, Enterprise Value, Total Cash Per Share, Profit Margin, Price/Sales, Book Value Per Share, Diluted EPS, Market Cap, Revenue Per Share, Net Income Avl to Common, Ticker, Date. Based on the date column, we can align the two datasets such that for each event in the S&P 500 dataset, we have the latest available financial report. From now on, we will refer to the dataset obtained from the alignment process simply as the *dataset*.

3.1. Feature engineering

We fill forward values whenever possible, we otherwise replace missing values with constant values (i.e. zero paddings). The intuition here is that if there are missing values at the beginning of the history of a stock, it means that the feature is not available, otherwise referring to the latest value as the most up-to-date one. As an example, consider holidays when markets are closed, and the price does not change since no orders are placed. We also fill the dataset with missing working days using the same forward-

fill strategy. Furthermore, we perform significant feature engineering steps adding technical indicators such as SMA and RSI. The SMA is computed by averaging the prices of a given number of multiple contiguous time steps. The RSI was, instead, computed using the relative strength formula (rsi).

$$U = max(0, closeNow - closePrevious)$$

$$D = max(0, closePrevious - closeNow)$$

$$RS = SMMA(U, n)/SMMA(D, n)$$

$$RSI = 100 - 100/(1 + RS)$$

It takes into account the differences in contiguous prices, with respect to the exponentially smoothed moving average. Therefore, it makes it easy to spot overbought and oversold events. In fact, using this technical indicator, we could extract the overbought and oversold binary features too. According to the RSI definition, a value that goes higher than 70 can be interpreted as a situation of overbought. Similarly, a value that goes below 30 is seen as a situation of oversold. When dealing with articulated models, there is not need to add such binary features since they can be easily extracted from the underlying data, the price in our case. Another step of the feature engineering process involved scaling the features to put them on similar scales. This step was very important considering that we are using stochastic gradient descent (SGD) to optimize the models and having features with completely different scales would have made the convergence task much harder. Therefore, we applied scaling to numerical columns such as: prices, volumes, technical indicator values, fundamentals, etc. At this point, we get all features to have zero mean and unit variance. Before training the model, we split the dataset into three subsets: *train set*, *dev set* and *test set*. These are respectively meant for training the model, tuning its hyperparameters and testing its performance. Since the core idea of the project is forecasting future events by looking at past data, we cannot leak future information in the train set. Therefore, we split the dataset by years. Having temporal data ranging from late 2003 to 2013, we dedicate the years from 2003 to 2011 to the train set, 2012 to the dev set, and 2013 to the test set. This way, the procedure is similar to backtesting the model strategy on present events. In order to predict the target adjusted close price, we consider the dataset according to the *sliding window approach*, also known as the *lag method*. In this way, we decompose the original dataset with overlapping windows and condition the target value only on its lag. During the training phase, we treat both the step size (i.e. the number of days before the next window) and the window size (i.e. the size of the lag) as hyperparameters and tune them accordingly.

4. Models

In this project, we apply the sliding window approach with deep recurrent neural networks. As our first models, we start with naive recurrent networks, respectively LSTM and GRU models. As opposed to vanilla RNN, LSTM and GRU networks partially solve the problem of vanishing gradient by employing a gating mechanism to regulate the amount of information to carry from previous time steps (Hu et al., 2018). The input data has the following shape: (*batch_size*, *window_size*, *feature_size*). Our initial idea was to build other components on top of the LSTM architecture in order to leverage some intuitions and assumptions that we had on the nature of our data. By comparing the performance of naive GRU and LSTM architectures, we experienced a quite noticeable improvement using the GRU over the latter. Therefore, we decided to build our modified architectures on top of the GRU. The GRU is a more recent alternative to the LSTM. In particular, it condenses the memory addition and deletion gates into a single one, which is just an input gate for the memory vector. This makes the GRU a simpler architecture that is much faster to train and that delivers better results in many use cases, especially with datasets of modest sizes. Furthermore, we assume that there exists a local pattern that we can leverage to predict future prices. In fact, in a third model, we make use of CNN layers to extract such information across time steps. It consists of a convolutional layer followed by a GRU layer, then connected to two dense layers. The convolutional layer is meant to extract high level patterns that form across time steps inside a window. This information is then also passed through the recurrent unit in order to get enriched by means of temporal sequencial correlation. As a fourth model, we employ the attention mechanism over our input data and then feed it to a GRU layer. In particular, we employ a multi-head self-attention mechanism to exploit the idea *to jointly attend to information from different representation subspaces at different positions* (Vaswani et al., 2017). Similarly to the previous model, here the idea is to process the raw input sequence by enriching it via a preliminary layer, and then extracting its sequential information via the GRU layer. However, in this case the attention layer operates differently compared to the CNN one. In fact, it works by putting the focus on specific parts of the input sequence. For this task, it can be useful since it can understand that specific subsequences of financial data are more meaningful than others and give those more importance.

5. Hyperparameters

In this section, we present a subset of the hyperparameters used. However, it is not intended to be an exhaustive list of all the hyperparameter. The list of runs is logged using wandb (Biewald, 2020) and can be reached at link.

Table 1. List of hyperparameters.

Hyperparameter	Value
Window size	20
Step size	1
Optimizer	SGD
Batch size	1024

6. Experimental results

As an evaluation framework, we consider multiple metrics in order to assess the quality of a particular model. Since we are dealing with a regression task, it comes natural to adopt the Mean Squared Error (MSE) measure as a proxy to indicate how well a model performs. Then, we measure the real performances of our system according to the return of a trading strategy. The system is designed to buy a fixed amount of stocks whenever the predicted price of the following day is higher than the current one. Then, it closes the position after one day and registers the gain or the loss. This way, the profit expectations can be compared among all the models. Another metric that we took looked at was the operation's accuracy. That is an unconventional metric that we decided to adopt to understand the fraction of times that the model predicted the correct trend. In other words, when the model predicted a price gain and the real outcome was actually a gain. Regression metrics alone were not enough to assess the goodness of the model in performing this task. In fact, a model could simply replicate the price of the previous day by applying the identity function and obtain decent MSE. This is because the relative change in price between two subsequent days is very small on average. However, such a model would make very careless predictions that would be of no help to a potential investor. Table 2 shows the quantitative results obtained with the models. We test the models on single stocks that were picked in the index. It can be seen that, in some cases, the models have considerably different results on different stocks. This reflects the fact that the trend of some stocks is more predictable than others. In fact, many different factors affect the dynamics of companies. For example, we notice higher return on AAPL that is a growth company, compared to GE which is instead considered more of a value stock. Growth stocks usually come with more volatility in the price, and this could justify this difference in profitability. Different models work better with different stocks. So, we could pick the model that maximizes profit on a given asset and use that to predict that particular asset. Considering the annual profit generated using our trading strategy, we can see that the predictions made by our novel architectures (i.e. GRU with self-attention and convolutional) outpeformed the standard recurrent approaches. In fact, either the attention-based or the convolutional GRU reached the very best results in those terms for a given stock.

Ticker	Architecture	MSE ($*10^{-4}$)	Operation Accuracy (%)	Year Return (%)
AAPL	Naive GRU	**5**	68	176
AAPL	Naive LSTM	773	**70**	107
AAPL	Attention GRU	79	64	213
AAPL	Convolution GRU	72	67	**306**
GE	Naive GRU	112	57	92
GE	Naive LSTM	158	55	113
GE	Attention GRU	**92**	**59**	127
GE	Convolution GRU	115	57	**133**
PFE	Naive GRU	91	61	149
PFE	Naive LSTM	172	60	143
PFE	Attention GRU	14	**75**	175
PFE	Convolution GRU	**10**	**75**	**193**
T	Naive GRU	14	61	84
T	Naive LSTM	32	59	50
T	Attention GRU	12	54	67
T	Convolution GRU	**7**	**65**	**85**

Table 2. Results comparison: MSE, operation accuracy, and year return using trading strategy.

7. Conclusions

We have seen a stock price prediction deep learning approach that brought several recurrence-based architectures to the table. At the beginning of this work we questioned ourself about the feasibility of such predictions that go against the efficient markets' theory. We demonstrated that a model is actually capable of extracting some patterns and use them to make profitable decisions. Thus, markets may have been not so efficient during the periods that are present in our dataset, especially in the 2013 test set. We would argue that the predictions made by these models could vary very much in profitability in different periods in time, affected by different events and circumstances. Furthermore, a potential widespread use of similar models by large institutions and individual investors could eventually annihilate their effectiveness, since the information extracted by them would be reflected in the prices. So, in order to keep the same levels of profit, possible alternatives could be more advanced architectures or more prior assumptions.

References

RSI – Relative Strength Index. https://en.wikipedia.org/wiki/Relative_strength_index.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Hu, Y., Huber, A. E. G., Anumula, J., and Liu, S. Overcoming the vanishing gradient problem in plain recurrent networks. *CoRR*, abs/1801.06105, 2018. URL http://arxiv.org/abs/1801.06105.

Lo, A. W. *Adaptive Markets: Financial Evolution at the Speed of Thought*. Princeton University Press, 2017. ISBN 9780691135144. URL http://www.jstor.org/stable/j.ctvc77k3n.

Mehtab, S., Sen, J., and Dutta, A. Stock price prediction using machine learning and lstm-based deep learning models, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Zou, Z. and Qu, Z. Using lstm in stock prediction and quantitative trading, 2020.